Addressing Syntactic Privacy for Privacy-Preserving Data Analysis and Data Release

A dissertation presented by

ROBIN ANKELE

to the

DEPARTMENT OF COMPUTER SCIENCE

in fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in the subject of

COMPUTER SCIENCE

SUPERVISOR	EXAMINERS
Prof. Andrew Simpson	Prof. Ivan Martinovic
	Prof. Chris Mitchell

Kellogg College, University of Oxford

Michaelmas Term 2020

Supervisor: Prof. Andrew Simpson Robin Ankele

Addressing Syntactic Privacy for Privacy-Preserving Data Analysis and Data Release

ABSTRACT

Existing approaches to tackle the challenges of privacy-preserving data analysis and data release are subject to vulnerabilities from certain attacks (which is the case for syntactic privacy models) or suffer in terms of efficiency, scalability or utility (which is the case for techniques based on secure multi-party computation). In addition, definitions of privacy (or any associated properties and notions) remain open to different interpretations among various stakeholders due to privacy's multi-dimensional and multi-faceted nature. In such environments, individuals who are not necessarily privacy experts, such as software developers or system designers, may struggle to select an appropriate privacy model or mechanism to protect their systems. This dissertation presents simplifications, analyses, considerations and promotions in the context of privacy-preserving data analysis and data release to support *utility*, *flexibility* and *privacy*.

As a first step, we facilitate understanding, application and analysis of syntactic privacy notions via abstraction to games. Via these games, we clarify understanding of, and relationships between, different privacy notions. Further, we give an unambiguous understanding of adversarial actions.

We analyse previously defined privacy games with regards to their applicability and relationships to each other, and define policies to support predominantly non-experts to establish an overview and to select the 'fitting' privacy notion / game for their applications. In this context, we utilise our game-based definitions to analyse and reason about privacy properties in a content-based clustering recommendation system as well as a collaborative-filtering based classification recommender system.

The second part is focused on the application to practice. Important in this context is the specification of requirements, which we derive from an analysis of multiple real world applications. Our use cases are predominately placed in distributed multi-party settings, where data remains split between mutually distrustful parties.

Given these real world constraints, we adapt and investigate a novel approach (based on trusted computing techniques) that remains resilient to many implementation-specific vulnerabilities, and increases efficiency and scalability. Our investigation comprises an advanced threat analysis covering high-level privacy model attacks to low-level side-channel vulnerabilities; furthermore, we present benchmarking results illustrating the superiority in performance of our approach compared to existing solutions.

Overall, we aim to foster understanding of privacy and applicability in data analysis and data release applications.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOTATIONS

Notations and Terminology

$\Pr$	...	Probability
$\mathcal{A}$	...	Adversary
$G_\star$	...	Game $\star$, where $\star$ is associated to a privacy notion
$\mathcal{M}$	...	Randomised Mechanism
$\mathcal{D}$	...	Domain (of a set)
$\mathcal{R}$	...	Range (of a set)
$\mathcal{X}$	...	Universe
ϵ	...	Privacy Budget

List of Mathematical Symbols

$a \circ b$	...	denotes composition of a and b
$a \parallel b$	...	denotes concatenation of a and b
$a \in A$	...	a is an element of set A
$a \notin A$	...	a is not an element of set A
a_i	...	i-th element (of a set)
$a_{i,j}$	...	j-th value of i-th element (of a set)
$\vec{a}$	...	denotes a vector of elements
$\vec{a} \backslash a$	...	denotes a vector of elements without element a
$\emptyset$	...	the empty set
$\lvert A \rvert$	...	denotes the number of elements of set A
$\lvert A_1 - A_2 \rvert$	...	denotes the distance between set A_1 and A_2 (*i.e.* the number of differing elements)
$a \leftarrow b$	...	assignment of element b (*i.e.* the value that b contains) to element a
$a \not\leftarrow b$	...	element a is not assigned to element b (*i.e.* the value that b contains)
$a \xleftarrow{\$} A$	...	uniform random assignment of a value of set A to element a
$f : A \to B$	...	mapping of an element $a \in A$ to an element $b \in B$ via a function f, where $f(a) = b$
$A \cup B$	...	denotes the union of two sets A and B
$A \cup \{a, b\}$	...	denotes the insertion of a tuple (a, b) in set A

$A \subset B$	...	set A is a proper subset of set B (*i.e.* there exists at least one element in set B, which is not element of set A)
$A \subseteq B$	...	set A is a subset of set B
$G_1 \vDash G_2$	...	denotes inheritance: G_1 may call procedures of G_2
$\Pr[\mathcal{A}^{G}_{\star} \Rightarrow \text{true}]$	...	denotes the probability that an adversary, $\mathcal{A}$, plays game $G_\star$, and $G_\star$ returns true
$\Pr[\mathcal{M}(D) \in S]$	...	denotes the probability that a randomised mechanism $\mathcal{M}$ using a dataset D outputs an element of a subset S
$\|\vec{a}\|_1$	...	denotes the L_1 norm of vector $\vec{a}$
$\|\vec{a}\|_2$	...	denotes the L_2 norm of vector $\vec{a}$

To Bettina and Walter

— who always supported me, dedicated their time through the ups and downs and pushed me to finish this dissertation.

ACKNOWLEDGEMENTS

First and foremost, I'd like to express my gratitude to my supervisor, Andrew Simpson. Throughout the last four years, you have been a source of guidance, support and advice. You've encouraged me to pursue independent studies of my own interests and topics which made this time most enjoyable. Thank you for all the patience (especially, while writing up in my last year), correcting all my spelling mistakes, and despite everything for letting me explore my research in privacy in my own time. I would also like to thank my advisor, Andrew Martin — for many interesting and fruitful discussions and his support over the last four years. Many thanks in this context also to Kubilay Ahmet Küçük — for our discussions alongside our aligned DPhils within the project AppTRE.

I would like to thank the Intel Corporation and the Department of Computer Science, University of Oxford for their generous financial support to allow me to pursue my studies. From Intel, I like to express my gratitude to Michael Steiner, Vinay Phegade, Carlos Rozas and Mona Vij for hosting us in Portland, US, and their guidance and our fruitful discussions.

Many thanks go to my office colleges with whom I shared not only a place of study, but also had many discussions about any topic outside of work. Thanks to Ranjbar (Ranj), Ahmad, Yudhistira (Yudhi), Kubilay, Pardeep, Ravishankar (Ravi), Justin and Andrew Paverd.

Many thanks go to all my colleagues from the Department of Computer Science and our research group including Chad, Eduardo, Martin, Majed, Yang and Emma. I'm extremely fortunate to have been able to work and collaborate with some fine scholars over the past few years, my co-authors and many others who I met through various conferences, workshops and summer schools. I'm happy to have had the opportunity to meet so many interesting researchers, to exchange ideas and share our time apart from the actual events.

Thanks to Kellogg College for providing accommodation, study spaces, delicious food at lunches, informal and formal dinners, letting me share the 'Oxford experience' and, foremost, for connecting me to so many amazing people who are my friends (too many to name all). You all enriching my experience, kept me distracting me over the years, but, foremost, I enjoyed every minute of your company. To name a few I would like to thank Vicky, Rob, Paritosh, Bastiaan, Alen, Bas, the infamous 'foosball group', Flo, Marc, and Ian.

More recently, I would like to thank William for letting me explore zero-knowledge proofs and supporting me financially while I was writing up my dissertation. To the team of Entrepreneur First, who taught me many invaluable startup and life skills, for their financial support, the encouragement and support to start my own startup and the necessary distraction which kept me engaged and motivated to finish while writing up. To Hasan and Mohamed, who took me under their wings — you have

taught me valuable skills in any directions from life skills to startup skills, and for valuing my expertise in cryptography.

None of this would have been possible without the support from my family and friends throughout my studies. In particular, I shall always be grateful to my parents, Bettina and Walter, for their unconditional love, support, encouragement and understanding. I'm indebted to gratitude to my twin, Ralph, with whom I not only share DNA, but, foremost, for the many discussions during my DPhil, his support and distractions and comments that improved this dissertation and for reminding me that there will be soon another Dr. Ankele.

Finally, I would like to thank Olivia for supporting me and motivating me to finish this dissertation. I am grateful for your distractions, making me laugh, being interested in my work and studies (and, sharing my interest outside of work as a foodie). You keep me motivated and pushing for our next chapter.

(I would also like to thank Southeastern Railways — for keeping a — not always steady — Internet connection on my daily commutes, that was used to write a good part of the final dissertation.)

Robin Ankele
September 3, 2019

I am thankful for the suggestions and comments by the examiners, Prof. Ivan Martinovic and Prof. Chris Mitchell, which improved the quality of the dissertation.

Robin Ankele
December 19, 2020

» *All human beings have three lives: public, private, and secret.* «

— Gabriel García Márquez in *Gabriel García Márquez: a Life*

1 INTRODUCTION

Contents

1.1 Motivation

In recent years, the field of data analysis and data release has flourished. This can be attributed to vast technological improvements and the development and availability of new applications for social, financial, medical and other day-to-day interactions between users and their (smart) devices. Such smart devices, all packed with sensors capable of collecting various kinds of data, are increasingly developed to collect, analyse and share data pertaining to our everyday life. Moreover, these devices are becoming increasingly interconnected (to each other and to the Internet), which enables data to be shared across the boundaries of homes, cities and countries.

As a result, a staggering amount of data has been created with estimates of 2.5 quintillion bytes of data created every day [238]. To put that into perspective, 90 per cent of the data that existed in the world in 2017 was created in just the previous two years [238]. Data is not only created via interaction between users and smart devices but largely also by users themselves.

> » *Data is growing faster than ever before and by the year 2020, about 1.7 megabytes of new information will be created every second for every human being on the planet.* « [302]

In various contexts, the collection of data can be beneficial in many ways and some new applications rely on accurate and vast collection of users' data. For example, advertisement systems rely on concrete demographic information about users to place targeted advertisements to those most likely to be susceptible to click and/or buy products that are advertised [294]. Similarly, in recommender systems information about the users' purchases and interests are of importance to be able to create new recommendations [46, 170, 368]. Within a smart home, information about human behaviour can be collected and devices can be switched on or off at certain times to

save energy costs, cause minimal disturbance to the users, and to provide increased comfort [438, 453]. In addition, data may be collected collaboratively over a larger set of individuals. Such sampling of population information has been proven to have the potential to reveal interesting new information that can be used in social, financial or medical research [414, 308, 299]. Even if a data collector has no clear application use case in mind, the utilisation of machine learning algorithms can reveal patterns in collective data.

Such use cases clearly point out the usefulness of vast data collections in our everyday lives. However, challenges may arise when processing sensitive data without compromising the privacy of the individuals whose records are contained in these datasets. In a broader context, privacy may be seen as the ability of an individual or a group to hide/suppress/seclude/restrict themselves, and/or information about them, from others[1]. The boundaries of what is private, however, may vary due to different social and cultural backgrounds, yet there are some shared common themes among them.

Over the years, various approaches have evolved in order to preserve an individual's privacy. Examples include the use of laws to:

(a) define information, pertaining to an individual, that requires privacy-preservation; and

(b) deter adversaries from accessing, manipulating or publishing such information.

To that end, data protection and privacy laws regulate against misuse of data such as the distribution to third parties, unintended processing and excessive collection and storage of data. Yet, most legal efforts have been directed to protect the privacy of an individual. For example, the European Union regulates efforts to preserve any 'personal data' of an individual via the General Data Protection Regulation (GDPR; Regulation (EU) 2016/679) [177]. Similarly, the Health Insurance Portability and Accountability Act (HIPAA; Pub.L. 104191)[2] regulates personal information of data that is 'identifiable' to a person within the US.

Similarly, approaches to limit someone's access to collective data may be applied. In particular, mechanisms of access control [7] can be used to regulate access and, consequently, mitigate unauthorised access to sensitive data of individuals. However, many of the previous approaches fail in case an application leaks (intentionally or unintentionally) any data and/or any significant statistical information about the data to the public. In particular, the vague protections ensured from the legal viewpoint (*e.g.* obfuscation of personal identifying information) have been proven to be insufficient in any computational setting if an adversary has access to auxiliary information about individuals involved in the data processing [338].

Despite the legal efforts to precisely define 'privacy' (or 'sensitive information' [8, 7]), various communities differ with regards to their definition of the concept of privacy. Concretely, privacy conceptions depend on different factors in each application and may vary between actors who contribute their data, or those who receive

[1] The used terms should be seen as a loose interpretation with the aim to identity the underlying principle: restriction of information from unwanted observers.

[2] http://legislink.org/us/pl-104-191

processed information. As such, the definition of privacy (or any related properties and notions) remains open to different interpretations among stakeholders within a community due to its multi-dimensional and multi-faceted nature. The complexity of defining 'what' is private and how different individuals perceive 'privacy' contribute to the lack of a commonly accepted definition of privacy (nor is there any realistic prospect of there ever being one).

The notion of privacy is also often confused with the guarantees that the underlying notions of confidentiality and secrecy aim to convey. While similar in intent, these notions aim for different goals: confidentiality means protecting the data of a system from access by unauthorised parties and secrecy means hiding of data from any party. On the other hand, preserving privacy of the data means that a system may reveal non-sensitive information about the data or release non-sensitive data, but does not leak any information about sensitive data. As such, privacy is, in addition, concerned with protecting any semantic information of the data. In other words, one could infer that confidentiality and secrecy are about protecting the data, while privacy is about protecting not just the data, but also the relationship between the data and an individual.

Nevertheless, in a computational setting, there is a strong incentive for a precise definition of privacy: the ability to verify (and/or to prove) that the output of an algorithm satisfies certain properties — in our case the preservation of an individual's privacy. On the one hand, definitions that are too vague (or properties that are relatable to privacy such as anonymity [8], participation hiding, pseudonymity [8] or unlinkability) may be subject to different interpretation and guarantees of the resulting algorithm. On the other hand, definitions that are too strict may lead to a limited environment in which privacy can be guaranteed.

Over the years various privacy notions have been proposed, became obsolete, been rejected or have evolved from previous ones. For example, in 1977, Dalenius [134] articulated a desideratum for privacy in statistical databases: an adversary should not be able to learn information about an individual that cannot be learned without having access to the database. Clearly, such a privacy guarantee would be desirable; unfortunately, though, Dwork [159] proved the impossibility of this desideratum. Various other privacy notions, for example [408, 298, 282], emerged and evolved due to earlier solutions' susceptibility to certain structural vulnerabilities. More concretely, in the absence of clear definitions for privacy and their formal treatment in a precise mathematical framework, practitioners have been tempted to apply various rule-of-thumb mechanisms to protect privacy. Examples of such policies include "reject query answers that include fewer than x entries" and "revealed results must include k-indistinguishable entries and pertain to certain value distributions within those entries" — see [187] for a detailed survey.

From the viewpoint of system designers, the expectation of the notion of privacy comes with different 'flavours'. Concretely, various concepts such as anonymity, unlinkability, undetectability, unobservability and pseudonymity may be considered as ways to characterise privacy. As such, privacy is not a universal definition, but, as a concept, involves various expectations or properties. Pfitzmann and Hansen [358] identified these 'flavours' and developed a concise terminology to unify and explore

relationships between these terms. In addition, Bohli and Pashalidis [72] similarly coin a terminology of privacy notions and show a hierarchy and relations between those notions. Concise phrasing and definitions of such 'flavours' and relationships of, and between, the same are essential for non-experts to gain a better understanding and abstraction of the concept of privacy as well as to relate said 'flavours' to real world objectives.

In this dissertation, we are interested in privacy aspects within the application scenario of data analysis and data release. Simply stated, data release is the process of releasing previously collected data from various sources, while data analysis transforms the collected data, before release, by means of any *analytic tasks*. There are many sub-variants of data release and data analysis, which we introduce formally later in this dissertation, for example: interactive, non-interactive and statistical data release and data analysis. In order to preserve privacy in these tasks, privacy-preserving mechanisms must be applied to avoid leakage of any sensitive information.

The importance of privacy considerations for data release and data analysis tasks can be motivated by some famous incidents where guarantees to preserve an individual's privacy failed. Examples include: the Netflix competition, where privacy researchers were able to re-identify individual users by matching the published anonymised data sets to the Internet Movie Database [329], the identification of an individual user in published pseudonymised search logs of AOL [49], and the case of the identification of William Weld, former Governor of the state of Massachusetts, by combining information from a public voter list and records in a published medical database linking through a combination of zip code, date of birth, and sex [408].

1.2 Research Problem and Contributions

There are many ways of preserving the privacy of an individual in a data collection via privacy-preserving mechanisms. However, one needs to take into account the inevitable trade-off between privacy and utility of any released information. While it is possible to achieve perfect privacy by revealing no information or revealing just random noise, this would not achieve any utility. As such, in the design of any privacy mechanism, a variety of factors to optimise for efficiency, utility and privacy need to be taken into account. Moreover, in real world environments, abstractions of idealised algorithm behaviour as well as assumptions about an adversary's knowledge do not hold. Therefore, privacy-protections need to be considered throughout a system and in all stages: starting from the data collection through the processing of the data to the final release.

What is more, non-experts, such as system designers and software developers, as well as beginners new to the field of privacy research, may struggle to establish an overview or to find appropriate privacy notions for their purposes, due to the previously discussed problems. As such, this dissertation is motivated by an aim to *simplify the analysis of privacy notions* to generate new insights and enhance the understanding of relations between privacy notions. In this context, we *support* current endeavours with software tools to foster engagement within the research community and to

bridge the gap to non-experts. We aim to *analyse* interesting use cases to motivate our contributions. Finally, we *utilise* novel software extensions with an existing trusted computing concept to define an alternative approach for multi-party applications. Our contributions enhance practical privacy considerations and guarantees in data analysis and data release contexts.

Within this context, this dissertation addresses the following research problem:

> *How can we engage and simplify the analysis of privacy notions and promote the adoption of techniques for efficient, flexible and accurate privacy-preserving data analysis and data release?*

The previous research question is split into four subtasks and address the following contributions:

(1) » *How can we simplify the analysis of current syntactic privacy notions?* «

Simplify. Game-based security notions have been widely used in the cryptography community to provide simpler and more easily verifiable proofs [55]. In Chapter 3, we adapt syntactic privacy notions via game-based definitions to simplify the analysis and generate new insights and relationships between those notions. The game-based definitions provide an alternative to the purely textual representation of other authors. Furthermore, our game-based definitions abstract the textual representation in algorithmic form to foster automated analysis of the privacy notions.

(2) » *How can we automate and engage experts and non-experts in the analysis of privacy notions and support the adoption of these insights in privacy-preserving applications?* «

Support. Experts and non experts typically engage with software libraries and study use cases to understand and utilise novel techniques in their settings and applications. In Chapter 4, for experts, we present an adversarial model, limitations and an analysis of two case studies via our previously defined privacy games. For non-experts, we present policies for the selection of a privacy notion and two case studies based on a content-based clustering recommendation system as well as a collaborative-filtering based classification recommender system. For both groups, we present our privacy games software library to: (a) foster analysis of privacy notions and (b) to engage non-experts in 'familiar' settings to adopt these principles to their applications.

(3) » *Which constraints evolve in real world settings/environments? What are the requirements and constraints to support the selection of efficient, scalable and privacy-preserving data analysis and data release?*«

Analyse. To gain understanding of the constraints and requirements within unfamiliar application settings case studies are often deployed and scrutinised to retrieve valuable insights. In Chapter 5, we present and analyse several case studies with respect to large scale real-world applications in a multi-party setting. From our analyses, we derive constraints and requirements for distributed, scalable and privacy-preserving applications.

(4) » *Which systems are capable of tackling the challenges of supporting efficient, scalable and privacy-preserving data analysis and data release?* «

Promote. In order to select a 'fitting' prototype for an application, system architects and software designers have to weigh up the trade-offs between privacy, utility, flexibility and scalability. In Chapter 6, we adapt and present an alternative approach to the traditional secure multi-party computation techniques. Our prototype is based on trusted computing techniques, remains resilient to many known implementation-specific vulnerabilities and is capable of supporting efficient and scalable privacy-preserving data analysis and data release operations. We test the relative efficiency of the two approaches via prototyping and show an advanced threat analysis and a performance analysis with regards to existing multi-party schemes. In certain settings, our approach precedes existing schemes with respect to the factors of privacy, flexibility and accuracy of privacy-preserving data analysis and data release operations.

1.3 Publications and Posters

This dissertation is based on research which was produced while investigating the previously stated research problem and has contributed to a number of papers. The following displays the papers arising from this work (in a non-chronological order) grouped by the contributions / chapters in this dissertation. Furthermore, we list posters in which we presented our work to the research community.

1.3.1 Research Papers

◇ Contributions from our abstraction of syntactic privacy notions to games (Chapter 3):

(1) Robin Ankele and Andrew Simpson. *Abstracting Syntactic Privacy Notions via Privacy Games.* In Proceedings of the 16th IEEE International Conference on Advanced and Trusted Computing (ATC 2019), IEEE, Aug 2019 [35].

◇ Contributions from our evaluation and analysis of syntactic privacy games (Chapter 4):

(2) Robin Ankele and Andrew Simpson. *Analysing and Evaluating Syntactic Privacy Games via a Recommender Systems Case Study.* In Proceedings of the 16th IEEE International Conference on Advanced and Trusted Computing (ATC 2019), IEEE, Aug 2019 [36].

(3) Robin Ankele and Andrew Simpson. *Extended Abstract — Analysis and Evaluation of Syntactic Privacy Notions and Games.* In Proceedings of the 16th Annual Conference on Privacy, Security and Trust (PST 2018), Aug 2018 [32].

- ◇ Contributions from our investigation and analysis of real world considerations and constraints (Chapter 5):

  (4) Robin Ankele, Kubilay A. Küçük, Andrew Martin, Andrew Simpson and Andrew Paverd. *Applying the Trustworthy Remote Entity to Privacy-Preserving Multiparty Computation: Requirements and Criteria for Large-Scale Applications.* In Proceedings of the 13th IEEE International Conference on Advanced and Trusted Computing (ATC 2016), pages 414-422, IEEE, Jul 2016 [34].

- ◇ Contributions from our evaluation and analysis of Trustworthy Remote Entities (TREs) in distributed settings (Chapter 6):

  (5) Robin Ankele and Andrew Simpson. *On the Performance of a Trustworthy Remote Entity in Comparison to Secure Multi-Party Computation.* In Proceedings of the 3rd IEEE International Workshop on Cloud Security and Forensics (WCSF 2017), pages 1115-1122, IEEE, Aug 2017. [31].

  (6) Kubilay A. Küçük, Andrew Paverd, Andrew Martin, N. Asokan, Andrew Simpson and Robin Ankele. *Exploring the use of Intel SGX for Secure Many-Party Applications.* In Proceedings of the 1st Workshop on System Software for Trusted Execution (SysTEX 2016), pages 5:1-5:6, ACM, Dec 2016 [273].

1.3.2 Posters

(1) Robin Ankele. *Output Privacy for Privacy-Preserving Data Release.* In Kellogg College — Bletchley Park Week Poster Competition. 2017.

(2) Olusola Akinrolabu, Robin Ankele, Ahmad Atamli, Ranjbar Balisane, Ravishankar Borgaonkar, Pardeep Kumar, Kubilay Ahmet Küçük, Yudhistira Nugraha, Piers O'Hanlon, Thomas Spoor, Tina Wu and Andrew Martin. *Trustworthy Systems.* In Oxford Cyber Security Open Day. 2017.

(3) Robin Ankele and Andrew Simpson. *Analysis and Evaluation of Syntactic Privacy Notions and Games.* In 16th Annual Conference on Privacy, Security and Trust (PST 2018), Aug 2018.

1.4 Dissertation Structure and Outline

1.4.1 Dissertation Structure

The structure of the dissertation is outlined in the following. We aim to present our contributions from theoretical to practical considerations. Each of Chapters 3-6 of this dissertation can be mapped to a contribution (presented in Section 1.2), and each contribution is informed by a research question (again, stated in Section 1.2). Taken together, the research questions frame the research problem that this dissertation aims to address. While research for this dissertation progressed, it gave rise to several research papers (outlined in Section 1.3). These research papers are the foundation

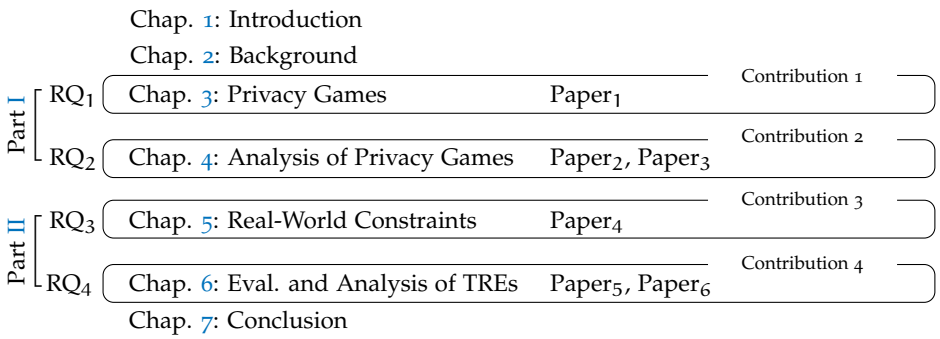


Figure 1: Outline of the mapping between research questions (RQ_x), chapters and papers via our contributions introduced in Section 1.2 of this dissertation.

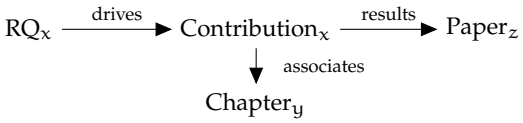


Figure 2: This graph indicates the relations of our contributions. As such, each contribution is informed by a research question (RQ_x) and drawn from research papers. The papers form the foundation of each resulting chapter of this dissertation.

upon which each chapter and, as such, each contribution is based. Figures 1 and 2 further illustrate said relations between research questions, contributions, research papers and resulting chapters.

1.4.2 Disclaimer on the Structure

This disclaimer clarifies how the following chapters are structured and why we have chosen to present it in this way / order. Overall, we are interested in the application of privacy-preserving data release and data analysis. Thus, we first introduce privacy (in a general context) because there is no commonly accepted notion or definition (of privacy): as privacy is multi-dimensional, multi-faceted, application and context-dependent, we aim to provide the reader with insights of the 'complexity' of reasoning about and preserving an individual's privacy. Then, we broadly introduce the application and context of data release and data analysis. In particular, we present current practice and methods to preserve privacy in such settings.

The main contributions of this dissertation focus on two of the major three privacy research strands[3]: (a) we introduce syntactic privacy notions and our representation of some of those notions in games to facilitate understanding of those notions; (b) we present an analysis and evaluation of the privacy notions and games to support non-experts to establish an overview / select a 'fitting' variant; (c) we analyse various use cases in distributed settings and derive requirements for large-scale applications. Finally, (d) we analyse the cryptographic approach of secure multi-party computation and introduce a method based on trusted computing techniques — both techniques operate in distributed settings and aim to tackle real world constraints (*e.g.* implemen-

[3]See Section 2.8.4 for details of the three strands of syntactic, probabilistic and cryptographic privacy.

tation and side-channel attacks, different trust assumptions and stakeholders). Moreover, we present an advanced threat model and a performance analysis.

Our approach combines strand (a) (focusing on syntactic privacy, and ensuring release privacy) with strand (c) (focusing on the distributed setting and, as such, ensuring privacy of the input values); we aim to promote a better understanding and simplified selection of 'fitting' privacy notions for the application of privacy-preserving data release and data analysis.

1.4.3 Outline

The remainder of this dissertation is organised as follows. In Chapter 2 we introduce the reader to the necessary background information to follow the contributions of this dissertation. As such, in Part I (of Chapter 2), we give a general introduction to privacy, discuss the complexity of finding an accurate and commonly accepted definition of the term, then introduce information privacy. In Part II (of Chapter 2), we introduce the reader to the tasks of data analysis and data release in a privacy-preserving fashion, outline existing privacy models, and state a preliminary threat model outlining attacks we need to protect against. Our contributions are split into two parts: Part I explores representations and modelling of privacy, while Part II leaves the 'idealistic' world and focuses on constraints arising from, and pertaining to, real world applications. Part I starts with Chapter 3 by addressing syntactic privacy notions. These notions impose a certain 'structure' on the released data in order to preserve privacy. To enhance understanding, we transform these notions into games. It turns out that these 'rules' are not sufficient to preserve privacy — for example, in the case of the presence of auxiliary information. Thus, it requires care in the selection of a 'fitting' privacy notion or game, which we address in Chapter 4. Concretely, we analyse our previous proposed games, introduce policies for the selection of a fitting game and evaluate the games in a case study based on recommender systems. Part II starts with Chapter 5, where we move towards practical considerations. In detail, we discuss the limitations of the idealistic world model and introduce real world constraints in privacy-preserving data analysis and data release. As such, we derive requirements abstracting real world constraints. In Chapter 6, we introduce a novel approach based on trusted computing, which enhances efficiency and scalability. Furthermore, we introduce an advanced threat model covering high-level privacy attacks as well as low-level side-channel considerations. We evaluate our approach in comparison with the current practice of secure multi-party computation schemes. Finally, we conclude in Chapter 7 and give pointers for future research.

» *Perfection is not attainable, but if we chase perfection we can catch excellence.* «

— Vince Lombardi

2 BACKGROUND

Contents

In this chapter, we introduce the reader to preliminary background information to help frame and motivate the contributions of this dissertation. This chapter is split into two parts:

Part I starts by giving a general introduction to privacy. This is followed by a discussion about the *complexity surrounding privacy* from a generic and computational perspective. Our focus in the latter parts of this dissertation is on *information privacy*. As such, we introduce terms relevant to privacy of personal information. The wording 'privacy' in this chapter, up to Section 2.3, shall be seen from a 'general scope'[1]. In latter parts, when we refer to privacy, we mean it as 'privacy of personal information'[2], unless otherwise indicated. We further discuss limitations and the kind of privacy guarantees that may be satisfiable in a privacy-preserving system. Before introducing the application scenario of this dissertation, we summarise the contents of this part. The terms and definitions used in this part follows loosely the definitions presented in the ISO/IEC 29100 standard [2, 8].

[1] When we refer to human aspects of privacy, we use the general scope.
[2] When we refer to *input, output, release, individual, group privacy* and *privacy-preserving systems* then we mean privacy of personal information.

In Part II of this chapter, we focus on the application scenario we are interested in: *privacy-preserving data analysis and data release*. We start by giving a general introduction to data analysis and data release and outline differences between these models. Next, we state privacy models developed in this context and discuss the differences between syntactic and probabilistic privacy models. Following that, we introduce generic operations, mechanisms and basic algorithms to achieve privacy under a given privacy model. Then, we elaborate on the privacy versus utility trade-off — more concretely, we present metrics which 'measure' the released information of a privacy mechanism; this information may be used to quantitatively assess the privacy/utility trade-off. Next, we introduce the cryptographic approach of secure multi-party computation and some preliminary information of Trusted Computing. Finally, we summarise the contents of this part. The terms and definitions used in this part follows loosely the definitions presented in the ISO/IEC 20889 standard [9].

> » *Many words have been written about privacy over the past year, including in these pages. I believe it's one of the most important topics of our time.* «
>
> — Sundar Pichai, CEO of Google

Part I — Defining Privacy

Privacy is a complex term. It has many meanings: for example, it is used to capture aspects of human behaviour (*e.g.* seclusion), but simultaneously it is used to describe properties of information exchange (*e.g.* computational aspects of privacy). This part shines light on some aspects of privacy (for the context of this dissertation).

2.1 Privacy

Privacy can take many forms, shapes, meanings, perceptions and expectations. It may be seen as a part of a human's nature, or something each human intuitively longs for in the presence of others. Historically, the concept of privacy has roots in philosophical discussions. An example is the well-known distinction between the two spheres of life by Aristotle: a public sphere (from Greek *polis* 'town') associated with political life, and a private sphere (from Greek *oikos* 'home') associated with domestic life. From the middle ages, the concept of privacy arose complementarily to the evolution of humankind with the availability and development of tools, technology and techniques. Examples include the use of internal walls in homes, silent reading in public spaces, solo beds and the use of envelopes or other techniques to hide information that is otherwise transfered in the clear.

Informally, privacy may be described as the ability of an individual or a group to seclude themselves, or information about them, from the observation of others. In this context, privacy is often associated with sensitive information inherent or pertaining to an individual. The concept of 'what is private' or 'what is not private' is typically

shaped by society and can differ among different cultures, but, in general, definitions share common themes. As such, the definition of sensitive information may include different properties among different jurisdictions, societies or individuals. An intuition of what categories sensitive information typically pertain to include: sex, race, personal income or earnings, political, cultural or societal affiliations, and physical or mental health condition.

Privacy is a concept or 'feeling' that most often only arises when there is something going wrong. As such, we can state a few questions (Q's) and answers (A's) that relate to when individuals typically consider privacy:

- Q: *'when' does it occur?*[3]
  A: In general, privacy comes to mind in the case of the *disclosure* of private (sensitive) information about an individual. Such an incident is often called or referred to as a *privacy breach* [8].

- Q: *'why' does it occur?*
  A: More and more tools and technologies are developed that 'ignore', exploit or do not provide the necessary means to preserve an individual's privacy and, as such, intrude or violate privacy guarantees.

- Q: *observation or protection by / from 'whom'?*
  A: In general, this may refer to literally any entity. Examples include governments, companies or any other individuals.

Privacy is a variable, inconcise and difficult-to-define 'measure'. In general, there are two types of expectations of privacy: *subjective* and *objective*. The former expectation of privacy refers to the belief and perceptions of an individual to experience or assume a certain state of privacy guarantees in the processing or observation of themselves. Subjective expectations greatly vary between different peoples' opinions and belief. The latter expectation of privacy refers to legitimate, reasonable and typical generally accepted beliefs and perceptions of privacy recognised by society and supported by privacy laws. As such, objective expectations are more formal and strictly defined (but may not cover every case).

Privacy advocates claim there is a 'right' to privacy. For example, in 1890, Warren and Brandeis [434] framed their definition of privacy, generally agreed to be one of the first publications advocating privacy in law, as a right to be let alone, to be undisturbed, unobserved, and/or free from public attention. Consequently, many constitutions and laws around the world incorporate privacy and/or define privacy as a human right. Another example, Article 12 of the Universal Declaration of Human Rights, states:

> » *No one shall be subjected to arbitrary interference with his privacy, family, home or correspondence, nor to attacks upon his honour and reputation. Everyone has the right to the protection of the law against such interference or attacks.*[4] «

What is more, the concept of privacy as a 'right' is, in general, accepted among western countries such as the United States of America and the member states of the

[3] By 'when', we mean the incident that leads an individual to consider privacy. By 'it', we mean the incident.
[4] Article 12, UDHR — http://www.un.org/en/universal-declaration-human-rights/

European Union. However, in recent decades there has been an ongoing discussion involving scholars and policy makers with regards to whether privacy should be defined as a separate right or should be part of already existing rights and laws. As such, laws regulating privacy may change over time to reflect the current technological and societal developments.

Privacy is studied in many fields. As a consequence, the vast literature has many different perspectives and viewpoints. For example, privacy from a legal viewpoint is concerned with defining laws to protect an individual's privacy. Policy makers use these laws to derive policies and principles within their organisations or governments. Philosophical research argues about the use, need and ethics of privacy, while historical research illuminates the 'evolution' of privacy over time. In the science community, privacy issues mainly arise due to technological advances. Concretely, the availability and development of new tools and techniques continuously create new privacy challenges. As such, in such contexts, privacy is studied from the aspects of protecting personally identifiable information [8] about an individual.

2.2 The Complexity of Defining Privacy

The term 'privacy' is frequently used in ordinary language. Yet, the complexity surrounding the term is manifold, preventing a single definition or commonly accepted interpretation of the same. In the following, we elaborate on 'what we see as privacy in a general sense' and 'what we see (or can achieve/satisfy) as privacy in a computational context'.

2.2.1 The Complexity of Defining Privacy in a General Context

We discuss the complexity of stating an 'accurate' definition of privacy with regards to: different interpretations, privacy as the bigger picture, legal ramifications, user behaviour, prior knowledge, and convincing the public. This list of points is not exhaustive; as such, there may be a number of additional points which we do not cover, but contribute to the complexity of defining privacy.

Different Interpretations. Over the years, research in privacy has gained much attention from jurists, legal scholars, philosophers, psychologists, and sociologists. The first informal definition of privacy follows the Warren and Brandeis law review article [434] of 1890:

Definition 1 (Privacy)**.** *Privacy is the right to be let alone, to be undisturbed, unobserved, free from public attention.*

With a few exceptions, this definition and related definitions of privacy within the current literature have been explored under distinct recurrent ideas. For example, Solove [403] identified six general reoccurring themes associated to core characteristics of privacy: (a) the right to be let alone; (b) limited access to the self; (c) secrecy; (d) control over personal information; (e) personhood; and (f) intimacy.

In a similar fashion, Jeffrey Rosen [376] connects privacy to: (g) the creation of knowledge; (h) dignity; and (i) freedom. Other 'synonyms' and rights surrounding or closely related to the concept of privacy are: (j) the right to be forgotten; (k) notice and informed consent; (l) anonymity; (m) confidentiality; (n) security; and (o) ethics.

These characteristics are, in general, all valid associations to the concept of privacy. Yet, various authors argue that these definitions stand alone and do not 'define' privacy. Concretely, Solove [403] contends the conceptualisation of privacy characteristics (a)–(f) from current literature as too narrow or too broad, and suggests that it would be helpful to understand privacy in terms of practices related to what we understand as private — the family, the body, and the home. Robert Post [362] argues that the categories by Rosen ((g)–(i)) are incompatible concepts with respect to privacy and should not be used to define privacy. To expand on this, various of the previously stated properties may be seen as 'means' or techniques in order to provide guarantees conveyed by privacy.

Privacy as 'the Bigger Picture'. As an alternative, these categories can be seen as a part of a (potentially overlapping, non-distinct) subset of what we understand as privacy. In other words, privacy may be seen as 'the bigger' overreaching concept, while previously stated 'privacy characteristics' are solely parts of the subset of privacy.

Other definitions related to privacy[5] contribute to the 'confusion' with regards to finding an accurate definition of privacy. While these definitions are somewhat more accurate and commonly accepted, the view on the meaning and value of privacy is still rather vague and can fluctuate between various stakeholders involved in any privacy process.

Legal Ramifications. Privacy is mentioned and regulated in the laws of many countries. Typically these laws protect the privacy of an individual from governments, private and public organisations, companies as well as other individuals in the context of storing, processing and releasing of personal identifiable information. International legal standards on privacy include: the APEC[6] privacy framework [37], the European Convention of Human Rights (ECHR) Article 8 [166], the OECD[7] privacy framework [336] and the International Covenant on Civil and Political Rights (ICCPR) Article 17 [420] of the United Nations.

More specifically, data privacy is regulated to protect personal identifiable information of an individual in a computational context. Examples of data privacy regulations include: the General Data Protection Regulation of the European Union [177] and the Health Insurance Portability and Accountability Act (HIPAA) of the United States of America [1].

[5] For example, *secrecy* is defined as the intentional concealment of information, *anonymity* [8] as the ability to conceal a person's identity [7], *confidentiality* as the controlled release of information to authorised persons, *security* as a collections of means or techniques to guarantee privacy, and *ethics* as the societal norms to reason about privacy.

[6] Asia-Pacific Economic Cooperation — https://www.apec.org/

[7] Organization for Economic Co-operation and Development — http://www.oecd.org/

While the GDPR [177] arguably represents the most strict regulations so far, data privacy in other countries is, in general, not highly legislated or regulated. The GDPR applies to all member states of the European Union, non-EU countries that aim to exchange data with EU countries, and companies in non-EU countries that want to do business with EU-based companies.

While it is important that there are laws and regulations in place, many of these regulations are not concrete or straightforward. For example, the GDPR underpins a number of principles of good practice for data privacy (Article 5, GDPR [177]): (a) lawfulness, fairness and transparency; (b) purpose limitation; (c) data minimisation; (d) accuracy; (e) storage limitation; (f) integrity and confidentiality; and (g) accountability. Nevertheless, there exist no concrete measures or guidelines of what techniques to use in order to protect an individual's privacy. As such, restrictions stated by laws or any measures achieving them are considered according to their reasonableness (which may or may not be sufficient to ensure strong privacy guarantees).

User Behaviour. The establishment of an adequate level of privacy protection also depends on the expectations and behaviour of users. For example, some users may not care so much about their private information in certain contexts and are willing to provide their data for a lower level of privacy guarantees in turn for a higher level of utility guarantees. In this context, Alan Westin conducted over 30 privacy-related surveys [274][8] between 1978 and 2004 from which he derived a categorisation (giving rise to a 'privacy index') of the public. All surveys were conducted via telephone and surveyed randomly-selected statistical samples of the adult population of the United States of America [274]. Westin's surveys measure attitudes and concerns about privacy: (a) on a generic level (*i.e.* broader privacy concerns of the public[9]), and (b) about specific privacy-related topics such as organisational handling of personal information, the use of medical records for research, and acceptance of a national identification system [274]. Importantly, Westin measured how these attitudes and concerns changed over time [274], but roughly stay within similar margins. Westin's so-called *general privacy concern index* (among various other indices) defined in 1990 study [274] suggests a classification of the public into the following groups: (a) privacy fundamentalists; (b) privacy pragmatists; and (c) the unconcerned.

- ◇ **Privacy fundamentalists** (about 25% of the public) are generally wary of providing their data to any processing tasks and are in favour of new laws and regulations to promote a user's privacy. In general, they choose privacy control over customer benefits.
- ◇ **Privacy pragmatists** (about 57% of the public) weigh their consumer benefits with the privacy provisions, but in a more relaxed way than the privacy fundamentalists. In their opinion, companies, organisations and governments should

[8] We acknowledge that the privacy indexes by Westin are dated. As such, the distribution of individuals pertaining to each category may fluctuate due to technological advantages and a thereof resulting changing society. Nevertheless, we argue that the categories are generic and are not going to change rapidly.

[9] For example, a survey question of Westin says: "Whether they agree strongly that business organizations seek excessively personal information from consumers?" [274]. Another question says: "Whether they agree strongly that the Federal government since Watergate is still invading the citizen's privacy?". [274].

have to 'earn' their trust to perform any processing. Finally, they prefer clear opt-out schemes in case of the involvement of their personal data.

- ◇ **The unconcerned** (about 18% of the public) are generally trustful with regards to how organisations collect and process their personal data. They do not advocate new privacy laws or regulations. Moreover, they forgo any privacy claims in order to secure consumer benefits.

In recent literature [441, 234], Westin's categorisation is criticised and it is argued that the categories may be poor predictors of context-specific user behaviour. As a consequence, other research has given rise to so-called 'privacy personas' [157], *i.e.* fine-grained clusters of users. However, while the levels (*i.e.* percentages) of the public fluctuate between different categories, may vary globally, and be context-dependent (*e.g.* biases in favour of particular countries or in favour of particular Internet-active individuals) all studies indicate that individuals show different preferences for privacy. Even though a categorisation of such preferences may not be easily quantifiable (and may not hold in many settings), it is that a system designer must acknowledge that such variation between users exists.

The field of privacy economics [17] further studies behavioural aspects of privacy (a deeper exploration is out of the scope of this dissertation and we refer to the relevant literature).

Prior Knowledge. By stating a definition of a privacy notion or privacy model one needs to consider which kind of sources an adversary may have access to in order to prevent such an adversary from simply breaking or invalidating the privacy guarantees. In an idealistic world, it may be assumed that an adversary has only access to the 'sanitised' version of a released database or the data analysis result itself. However, such thinking can not apply in a real world setting, where an adversary may only be limited by physical laws and not by assumptions we make.

In this context, background knowledge or auxiliary information about a dataset plays an important role. Auxiliary information may be defined as any additional or complementary information about individuals who are participating in a data processing task. Then, privacy may be easily breached if an adversary is able to connect this auxiliary information (typically public information from multiple sources) to the dataset. O'Hara described and termed this kind of combination of multiple sources in [337] as 'jigsaw re-identification'. Examples of such kind of privacy breaches include [188, 329, 49, 408].

What is more, these sources may be private in the case of an isolated viewpoint; however, they may reveal private information when combined. Thus, it is recommended to utilise privacy models which are not vulnerable to such attacks. An example would be any variants of differential privacy [159].

Convincing the Public. It is one thing to design a privacy-preserving system and another to convince the public that such a system does indeed preserve an individual's privacy. Such a discussion essentially boils down to two things: a generic trust of people in systems (which may be drawn from the categories of Westin [274]) and

the understanding of the privacy guarantees or properties such a system is trying to convey.

To add to the former, convincing privacy fundamentalists and pragmatists may require the establishment of a trust relationship. As such, if the system is able to act in the expected manner for multiple runs over a certain period of time, then these participants may be willing to provide their personal data. Clear formulated privacy goals, methods and techniques to achieve them are essential in that case.

To add to the latter, compliance with laws and regulations to provide certified services may be helpful. Many of the current acceptable techniques for information privacy rely heavily on mathematics (*e.g.* differential privacy [159]). As such, these guarantees must be formulated in ways that the general public is able to understand.

2.2.2 The Complexity of Defining Privacy in a Computational Context

In contrast to the previous definitions, privacy in information systems is mainly based on the processing of data pertaining to individuals. However, privacy in computational environments is not a universal definition, but also comes with different kind of 'flavours'.

Informally, we define *information privacy* within the context of this dissertation[10] as follows:

Definition 2 (Information Privacy)**.** *Information privacy is the relationship between the collection, retention and dissemination of individuals' personal data and the individuals' interests in controlling*[11]*, or at least significantly influencing, of said handling in a computational setting.*

Although there is a clear distinction between information and data (as we shall see in Section 2.3), in the literature, both terms are used simultaneously. To follow this convention we say information privacy is concerned with regards to information that is 'held' by the data. This includes information about the data and any meta-information about the data. Information privacy is defined in accordance with a computational setting — as such, we associate the application domain to those which include computational devices.

Information privacy, as defined in Definition 2, is informed by the seminal privacy definitions of Roger Clarke [115]: (a) as a measure of the privacy interests per user, it is defined as a variable value (and differs between users); (b) as a property of a system, it is defined as a binary value (assuming a defined, application-specific threshold) — (i) the system is information-private or (ii) the system is not information-private. Privacy *interest*, as defined by Clarke, has multiple dimensions such as: (a) privacy of the person; (b) privacy of personal behaviour; (c) privacy of personal communications; and (d) privacy of personal data. In this context, information privacy is mainly associated to a combination of dimensions (c) and (d). A different way of describing

[10] Due to the different forms, shapes, meanings, perceptions and expectations many definitions of privacy have been proposed. Generic privacy definitions that cover all aspects are out of scope of this dissertation. Our privacy definitions are limited and associated within this dissertation's underlying application use cases.

[11] Under the term 'controlling', we understand the expression of an individual's interests, to their best ability, in order to control/influence the outcome of the task under question.

the relationship of these 'competing' interests is in terms of disclosure of information (see Appendix B for definitions of disclosure types and information gain). While an individual's interests may be to limit the amount of disclosure about their information, an analyst's aim may be to increase their information gain.

Information privacy can be further classified: (a) by means of the 'processing' function that is evaluated on the data; and (b) by the way data is released to the public. We refer to the former as *computational* or *processing privacy* and to the latter as *release privacy*. Both types are subsets of information privacy and exclude the collection and retention phase. In other words, these sub-types apply to the data processing and release stages. Computational privacy, on the one hand, is concerned about the privacy interests during the step of *data processing*. Release privacy, on the other hand, is concerned about the dissemination of data after processing. (Storage privacy, another example, *i.e.* when data is at rest, is ruled out of scope and, therefore, not addressed in this dissertation[12].)

Informally, computational privacy within the context of this dissertation may be stated as follows:

Definition 3 (Computational Privacy)**.** *Computational privacy is the relationship between the processing of individuals' personal data and the individuals' interests in controlling, or at least significantly influencing, said handling.*

Definition 3 is concerned about the competing interests of an analyst to process individuals' data for, for example, the purpose of extracting meaningful information (typically to pursue their business interests), whilst an individual's interests are to retrieve 'value' in consenting to providing their data for such processing.

Consent [8], *i.e.* permission, to execute certain processing functions for a specific purpose [8, 7] and the restriction of limiting the number of parties (*e.g.* individuals or computational devices) that have access to the data is typically asked in form of privacy notices [8]. In this context, consent comprises two important properties: (a) restriction of processing purpose [8, 7]; and (b) restriction of processing access by third parties.

Consent, without the restriction of the processing purpose, would lead to analysts performing any 'processing' functionality. For example, this means that a clever (and potentially malicious) analyst may be able to extract sensitive data by processing certain tasks on the data provided by a naïve data owner, while fully satisfying the legal requirements. A blatant example of a malicious function would be to just release all input values of the data owners without any obfuscation to provide privacy. In other words, an analyst may perform malicious tasks that a data owner is unaware of and hasn't given consent to.

Consent, without the restriction of the processing access by third parties would allow an analyst to widely distribute the data without any limitations.

While computational privacy is a good start (considering well defined consent statements), one case that is not covered, but very relevant in modern computing, is the leakage of meta-information. In other words, computational privacy does not cover the case if information of an individual is leaked from the processing function's result (*i.e.* the outputs of the processing function). For an illustrative example consider

[12] Some rights and restrictions of 'storage' privacy are addressed by the GDPR [177].

the following: an analyst aims to assess the number of people who smoke within a dataset. The outcome of a count function may be considered computationally private, but applied just before and after the addition of a person who smokes reveals that this person, indeed, is a smoker.

Privacy-preserving systems which only satisfy computational privacy are therefore not sufficient to guarantee/optimise an individual's privacy interests. Privacy notices [8] and consent are one way to inform the user of the purpose of processing and if third parties are involved in such processing. (A better, more privacy-preserving way may be to request a user consent whenever a specific function (associated to a specific purpose) is triggered and/or a third party is involved. Such a request-based permission for consent[13] to process is out of scope of this dissertation.)

To address the leakage of information from the processing output, we informally state the definition of *release privacy* within the context of this dissertation:

Definition 4 (Release Privacy)**.** *Release privacy is the relationship between the release of processed information, the 'outcome', that was generated by processing individuals' personal data and the individuals' interests in controlling, or at least significantly influencing, said handling.*

Computational privacy and release privacy are defined as variable, competing measures (between the interests of an analyst and the individuals involved in such processing). Moreover, given an application specific threshold, a function can be defined as 'release-private' if the 'outcome' of the processing does not leak any information (beyond that specific threshold) that can be traceable to the input values (*i.e.* personal information) of an individual.

Assuring the 'protections'/'rules' outlined by computational privacy and release privacy may ensure (or, at least, is a good starting point to ensure) privacy-preserving processing of personal data of an individual. As such, on the one hand, computational privacy aims to protect any sensitive input values of the computation. On the other hand, release privacy aims to ensure privacy protection of the released output.

2.3 Information Privacy

Information privacy is the relationship between the collection, processing and storage of information [7], and the expectation and preservation of an individual's privacy. Overall, we consider information privacy to be an aspect of privacy. As such, it is limited to reasoning about privacy in the context of the processing of data in any computational system. Nevertheless, this is still a very broad definition. Thus, in the following, we outline some basic terminology including: data, data subjects and operations on the data[14].

In a computational system, data is typically represented by data structures and stored on the device in binary form. As such, data, informally, is the physical rep-

[13] For example, see Apple's user permissions (https://developer.apple.com/documentation/uikit/protecting_the_user_s_privacy/requesting_access_to_protected_resources) system and app entitlements (https://developer.apple.com/documentation/bundleresources/entitlements).

[14] These definitions loosely follow those of the General Data Protection Regulation [177].

resentation of any information which can be processed via a computational system. Information is what an observer learns by interacting with the data. In other words, information is the 'meaning' of the data. In this context, information also includes any meta-information (*e.g.* semantic information) about the data[15].

Nevertheless, when it comes to talking about privacy, the phrases 'information privacy' and 'data privacy' are typically used simultaneously. From the previous distinction, however, data privacy refers to the sole protection of data. In contrast, information privacy refers to protect data and, in addition, to protect any meaning of these data.

Information privacy issues typically arise in the processing of 'personal' data (especially in the processing of 'sensitive personal' data[16]). The former refers to any data which relates to an individual who can be identified from those data or from those data and any other information which comes in the possession of a data controller. The latter refers to specific categories of personal data (similar to those discussed in Section 2.1: sex, race, personal income or earnings, political, cultural or societal affiliations, and physical or mental health condition). Data which falls into these categories are more likely to be of a private nature, as it may be used in ways that are discriminatory. Thus, those data must be treated with greater care than other 'non-sensitive' data.

In the processing of information several stakeholders are involved including: data subjects, data controllers and data processors. In short, a data subject is the entity whose data is processed, data controllers decide on the purposes for processing and data processors act on the behalf of a data controller. Information privacy is defined to ensure 'privacy-preserving' processing among such stakeholders and their data. Data records may originate from various sources; as such, information privacy may be categorised into subgroups including: Internet privacy, medical privacy and financial privacy to name just a few.

Historically, information privacy has evolved with the development of personal computers and, subsequently, has been adapted regularly in response to new improvements and developments of technology. As such, information privacy was first considered in the statistics community in the context of publication of data from census. For example, in 1977, as we have already discussed, Dalenius [134] stated a desideratum about the acceptable knowledge gain of an adversary with access to a statistical database. In data communications, privacy is studied with respect to data transmissions. For example, in 1981, Chaum [103] laid the groundwork for the field of anonymous communications by proposing so-called *mix networks*. With the continuous improvements in the field of databases, data mining and data analysis, the Computer Science community started facing similar problems and engaged in research to consider privacy. As such, in 2000, two contributions in the field of privacy-preserving data mining were published. Agrawal and Srikant [24] presented general methods

[15] There is a subtle difference between data and information. To clarify, data are raw, unorganised facts which, when seen individually, are rarely useful. When data is processed (*e.g.* organised, structured, presented in a given context to make it useful) information is gained. Thus, the relationship between data and information is: data are unorganised pieces and, when processed, information is derived from the data.

[16] We note that privacy issues may also arise when processing non-sensitive personal data, which may result in a privacy breach [8]. (Such a breach may be even more devastating, since, typically, non-sensitive data is treated less securely.)

to hide sensitive information, and Lindell and Pinkas [289] showed the application of cryptographic tools to preserve individuals' privacy. Over the last two decades, several privacy models have been defined to establish when it is 'safe' to release data and when data release leads to any form of disclosure. We discuss some basic privacy models in Part II of this chapter and explore more complex ones in Chapter 3.

2.4 Limitations and Satisfiability of Privacy Guarantees

Privacy definitions may be used to regulate the information leakage of a privacy-preserving system. Different definitions may guarantee different levels of privacy, which in turn can be used to regulate the privacy/utility trade-off. In the following, we informally discuss the complexity of setting an appropriate privacy level (typically defined by a parameter of a privacy notion) and what kind of privacy guarantees may be satisfiable within a computational system.

The optimal guarantee of privacy is *perfect privacy*, which is studied by various authors including [369, 91]. This notion requires that no private information is leaked in the process of releasing data. While such a guarantee may be desired in the context of guarantees for privacy, it may not be preferable in the context of data utility. For example, perfect privacy can be achieved by not revealing any information at all or revealing just random data. More concretely, perfect privacy may be achieved if certain requirements as outlined in [369] are satisfied. Thus, such a release would either give no data utility at all or only very limited data utility in a strictly restricted setting.

Another consideration for privacy guarantees is the capabilities of an adversary. In particular, an adversary may be computationally unbounded (as such, the privacy-preserving system must guarantee privacy in an information-theoretic setting) or may be computationally bounded. The former type guarantees strictly stronger privacy, while it may be also harder to satisfy protections against such an adversary. The latter type implies some assumptions on the capabilities, which may or may not be satisfied by the threat model of a particular application. Examples of restricted adversaries for privacy notions include [349, 319].

Various privacy models require the system designer to set a parameter to guarantee varying levels of privacy. This parameter typically controls the privacy/utility trade off of the release data. Examples of such privacy models include [408, 298, 282, 159]. Nevertheless, it is often not completely clear how to set such a parameter. Setting of such parameters are often inclined to various applications or context-specific problems and may not always be generalisable. As such, the setting of such parameters requires expert input. A prominent example is the setting of the ϵ parameter for differential privacy [159] — [235, 280] provide an initial approach to clarify understanding of the parameter and how system designers may be able to choose an appropriate value.

It is important, for system designers, as well as the general public, to realise what privacy guarantees can or can not be achieved by a privacy model in a computational system. Within the differential privacy community this discussion [264, 313] has given rise to a distinction between 'your secrets' and 'secrets about you'. The former, 'your

secrets', pertains to actual information that only comes from your participation in a (computational) task. The latter, 'secrets about you', pertain to information about the general population that *may* also applies to you, even without your participation in a data-processing task. The keyword *may* here depends on any auxiliary information an adversary is able to collect, combine and analyse about you from any other sources. As such, the ability of an adversary being able to gain knowledge from the observation of a data release and the combination with other public information is mostly independent of the actual information that you provide in a data analysis process. As a consequence, modern design guidelines for privacy definitions consider 'your secrets' as privacy, and 'secrets about you' as forgettability.

What can be concluded from the above discussion is that there exist many definitions of privacy as well as various privacy models abstracting such definitions. Nevertheless, a computational system is somewhat limited in what it can achieve and, as such, not every privacy definition is achievable. Privacy guarantees rely on the appropriate setting of certain parameters, yet, often this setting is unbounded or only vaguely defined by the designers. Thus, when designing a privacy-preserving system, one must be clear as to what guarantees the system is able to achieve.

2.5 Summary

Privacy can take many forms, shapes, meanings, perceptions and expectations. Expectations and perceptions of privacy have evolved over time as we have seen. Conceptions of privacy differ between societies and cultures. Furthermore, expectations of privacy may be categorised as subjective (*e.g.* how an individual perceives privacy) or objective (*e.g.* how privacy is recognised by society and under the law). Privacy is a variable, inconcise and difficult-to-define 'measure'. Although it is recognised in many laws, there are variations of interpretations of the same and ongoing discussions by scholars, policy makers, philosophers and jurists.

The term 'privacy' is frequently used in ordinary language. Yet, the complexity surrounding complexity of the term prevents a single definition or commonly accepted interpretation of the same. In particular, the complexities arise from different interpretations of the term, privacy being a superset of other privacy characteristics, different legal perceptions and definitions, manifold user behaviour and the dilemma of the existence of background knowledge. Nevertheless, in more restricted settings, such as information privacy, informal definitions exists. For example, computational privacy is concerned with the preservation of an individual's privacy, while the processing of a function and release privacy is concerned with the preservation in the context of releasing 'anonymised' versions of the dataset or privacy-preserved query responses.

We have introduced the reader to concepts and terms of information privacy, which we will be using in the following part and chapters. Our terms, in this context, draw from the GDPR [177] as well as other sources.

We have seen that there are limitations in the privacy guarantees that may be achieved by a privacy-preserving system such as 'your secrets' and 'secrets about you'. Further, in an 'idealised' world, assumptions about potential adversaries are made,

which do not always hold in 'real world' settings. Thus, it is important to recognise limitations of privacy in some contexts.

So far, in this part, the reader has seen that there are a lot of factors that contribute (or are to be considered) for a 'statement or definition' of privacy. In the next part, the focus is on applications which are concerned with protecting the privacy of an individual. Concretely, the reader is introduced to the application and techniques of privacy-preserving data analysis and data release. Then, in Chapters 3 and 4, we set out to simplify and support the selection of a 'fitting' privacy notion for tasks defined in the next part, while taking in consideration the definitions and limitations of this part.

» *When we do collect data, we will use it to benefit you and to make your experiences better..* «
— Satya Nadella, CEO of Microsoft

Part II — Privacy-Preserving Data Analysis and Data Release

The extraction of meaningful information from large data collections is a major aspect of modern computing. Information is power. But, power comes with responsibility. Preserving the privacy of individuals and groups in any data analysis process is a key factor of this responsibility. Privacy drives trust; trust drives adoption; and adoption provides access to information. In this part, we elaborate on aspects of modern privacy-preserving data analysis and data release (for the context of this dissertation).

2.6 Data Analysis and Data Release in the Plain

In order to extract meaningful information from large data collections techniques of data analysis and data release may be applied. In the following, we motivate the approach of analysing data and data release. Next, we describe the difference between these two models and the stakeholders involved in such tasks. Furthermore, we discuss design considerations of two natural modes: non-interactive and interactive data analysis / data release. We outline popular types of data analysis algorithms and, importantly, introduce techniques to preserve an individual's privacy for the given models.

2.6.1 Motivation

Algorithms and approaches to analyse vast collections of information can be collectively described as knowledge discovery or data mining [111, 180]. These approaches can broadly be distinguished into:

Definition 5 (Data Release)**.** *Data release is a process which takes unstructured collective information about a collection of individual(s) and reveals it for the purpose of further analysis. While this step is performed on a system, A, any (structural) processing of data is performed within a separate system, B, or within a separate process, C.*

Definition 6 (Data Analysis)**.** *Data analysis is a process which transforms (and reveals) unstructured collective information about a collection of individual(s) into structured information, where the processing and release steps are done within the same system or process.*

Definition 5 is concerned with the *release* of unstructured information as a means of a 'middle' man which, for example, in the context of this dissertation, preserves the privacy of the individuals, but any structural processing and analysis is performed in a separate system or process. Concrete examples are stated in the following paragraphs. On the contrary, Definition 6 is concerned with the *analysis* of unstructured informa-

tion in a structured way and *includes* the release processes within the *same* system or process.

Whist being similar, these processes have different application scenarios. Examples of data release may include: governments and organisations providing information such as patient records, business and economic data, environmental data, transport data and societal data. Data is often released to ensure transparency of a government's / public organisation's operations and actions or for any other research or business purposes. Examples of data analysis include: e-commerce personalisation, log analytics, recommendation engines and fraud and crime prevention. Data analysis techniques are typically used in marketing / retail, finance or manufacturing industry or organisation and governments to analyse big chunks of data and reveal interesting patterns within those data.

2.6.2 The Model

In the following, we introduce the model of data release and data analysis with regards to: multi-step data release and analysis process, stakeholders and generic design overview, architectural design (*i.e.* distributed usage models), and query design (in terms of non-interactive vs interactive release and analysis).

Multi-step Data Release and Analysis Process. The tasks of data release and data analysis are not one-off operations, but typically involve an iterative sequence of steps [219]. Broadly classified, data may be collected, pre-processed, selected and transformed before being released or further processed by a data analysis algorithm. The following outlines these steps in more detail (in this listing we omit architectural details and focus solely on the operational tasks):

Data Collection: (1) Data Collection [7]

Pre-Processing: (2) Data Cleaning, (3) Data Integration, (4) Data Selection and (5) Data Transformation

Data Release or Data Analysis: (6) Data Release or Data Analysis

Post-Processing (typically only applicable for data analysis): (7) Pattern Evaluation and (8) Knowledge Representation

Data release tasks typically involve steps (1)–(6), where the data is thereafter released to a data analyst for any further processing. Data analysis tasks typically involve steps (1)–(8), however, some steps may be skipped and/or additional steps added depending on the specific data analysis operation.

Stakeholders and Generic Design Overview. The process of data release or data analysis is a multi-entity process, which involves several (typically separate and distributed) entities.

Figure 3 outlines the difference between both modes. While the data collection phase is the same for both modes, the left-hand side of Figure 3 illustrates the data release process and the right-hand side of Figure 3 illustrates the data analysis process.

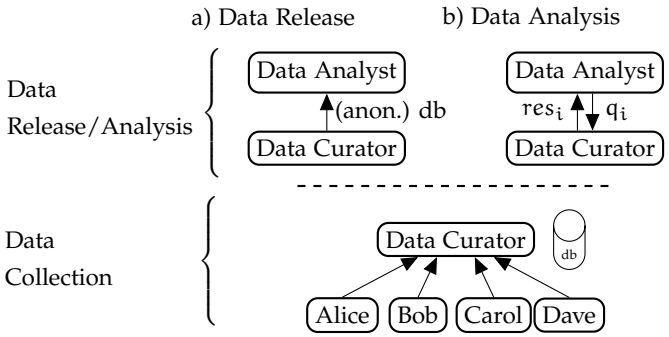


Figure 3: Illustration of the differences between data release (a) and data analysis (b) modes. After an identical data collection phase, in the case of data release, data is released by the data curator to the data analyst as a whole (*e.g.* anonymous dataset). In the case of data analysis, the data analyst is allowed to submit queries q_i to the data curator and is given responses res_i.

In this context (for now, and without considering any architectural design decisions), we assume a single data curator[17], a single data analyst[18] and multiple record owners[19]. (Chapters 3 and 4 mostly follow this convention. Cases of multiple data curators and multiple analysts are discussed in the next paragraph and in Chapters 5 and 6.) In the *data collection* [7] phase, a data curator collects data from record owners (*e.g.* Alice, Bob, Carol and Dave). This phase is identical for both modes of data release and data analysis. In the data release / analysis phase:

(a) in data release mode, the data curator shares the collected records as a whole to the data analyst;

(b) in data analysis mode, the data analyst is allowed to submit queries to the data curator, who, in turn, processes this query and responses with the query answer.

In other words, in the data release mode, the 'processing' of information is located *outside* and/or separate to the current process at the data analyst — the data curator is only collecting and releasing the records. Or, simplified: the curator acts as controller and the analyst as processor [8, 177]. Any released information is final and, as such, can't be modified. If data is subsequently added to the data curator, a completely new release must be prepared. In the data analysis mode, data is processed directly at the data curator. Again, simplified: the curator acts as controller and processor. Consequently, the records owner's data never leaves the data curator. Over time the data may be modified or deleted. Any data analyst only gets query access to the data

[17] A *data controller* or *PII controller* is an entity who decides on the processing function that is applied on the collected data. As such, this entity is responsible to ensure that the selected processing function is, for example, compliant with good practices for data privacy and, as such, does not leak information about a data subject's sensitive data. The data controller may also act as a data processor [8, 177].

[18] A *trusted analyst*, *data processor* or *PII processor* is an entity who 'executes' the processing function that a data controller has selected. In case these two are separate entities, the data processors acts on behalf of the data controller. A data processor is, in general, also responsible for ensuring privacy-preserving processing [8, 177].

[19] A *data owner*, *record owner*, *data subject* or *PII principal* represents an individual whose data is processed. The data provided by such an individual may be sensitive or non-sensitive. This data may also be associated to an unique identifier, which can link the identity [7] of such an individual to the data [8, 177].

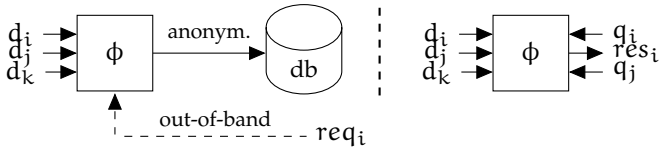


Figure 4: Illustration of the two natural data release and data analysis modes: non-interactive (left) and interactive (right). A system ϕ receives data records d_* from record owners and allows a data analyst to either submit requirements req_i via an out-of-bands channel to release a dataset db (thus, non-interactive) or answers to queries q_* with responses res_* (interactive).

curator through a pre-specified interface. To this end, all entities trust each other and are fine with sharing their information. In Section 2.7 we discuss cases of varying trust assumptions of the data curator and data analyst in more detail.

Query Design: Non-interactive vs Interactive Release and Analysis. There are two natural modes of how a (or multiple) data analyst(s) interact with a data release or data analysis system. We consider the interaction with the system to be either *non-interactive* or *interactive*. Figure 4 illustrates both modes.

In the non-interactive setting, we assume data is to be shared by a data curator, who publishes a 'selected' version of the database. With the wording 'selected' we mean trimmed to the data analyst's requirements. These requirements are communicated either beforehand to the data curator, are specific to the release system in general, or are communicated via an out-of-bands channel. In this mode, data is typically released as a whole for the purpose of further analysis by a data analyst. As such, non-interactive data release is preferred in settings where data is collected over a given period of time and thereafter released. Data which is added to the system after a release may only be included in a different batch of released data. This impacts also how data may be maintained (*i.e.* modified, deleted and added). Since a released batch may not easily be withdrawn or modified, these consideration must be taken into account when designing such a system.

A data analyst interacts with such a system as a 'passive' entity. It may communicate their requirements over previously mentioned channels, or depending on the system's purpose, receive a 'fixed-styled' dataset.

In the interactive setting, we also assume data is to be shared by a data curator; however, in this case, the data remains with this entity. A data analyst is typically given an interface via which she is able to submit queries to the data curator. In turn, the data curator executes these queries on the dataset and returns the responses to the data analyst. The data analyst may be limited in the amount of queries which are submitted to the system as well as the type of queries. Since the data remains with the data curator, any record owner can easily submit a query to modify, delete or add any additional information.

A data analyst interacts with such a system as an 'active' entity and is, in general, allowed to submit multiple queries. It may be distinguished if the queries can be submitted in an adaptive fashion (*i.e.* the data analyst receives a response to a query

and can modify any future queries according to the received data) or if the queries are fixed before being sent to the system.

The relationship between the modes of data release and data analysis and the query 'style' of non-interactive and interactive querying may be easily confused and, as such, falsely associated with each other (in the given order). To clarify, the modes of data release and data analysis refer to the operational task of a system. In other words, it refers to a system's 'purpose' and describes 'how' a system interacts with its participants. Concretely, non-interactive and interactive querying refers to how a stakeholder (*i.e.* in our scenario, a data analyst) interacts with the system. Further, it is possible to combine the previous introduced modes via any combination (*e.g.* non-interactive data analysis, interactive data release); yet, only some combinations are preferable (due to efficiency, scalability and flexibility).

2.6.3 Requirements and Challenges of Data Release and Data Analysis

While recent literature has produced a lot of promising research, there are still a number of challenges that need to be addressed in order to apply these approaches in real-world deployments:

Privacy and Data Security. When data can be viewed from different angles and various abstraction levels through analysis via data mining algorithms, previously unidentified patterns may be revealed, which may threaten the privacy of individuals who contributed their data in such operations. It remains important to study when knowledge discovery leads to an invasion of privacy and, consequently, put forward countermeasures to avoid such leakage. Fortunately, some of these issues get addressed (at least, in part) through considerations such as privacy by design [100] and newly adopted data protection regulations [177]. What is more, a whole field of research is dedicated to addressing these kind of issues (see Section 2.7). Another rarely addressed problem closely related to privacy is the trustworthiness of the underlying computational system (*e.g.* software stack, operating system, hardware). In general computational systems, there are no guarantees that the owner of the hardware is not malicious and will not try to intervene in the data analysis operation. From the software side, there are no guarantees that the operating system or other malicious applications are observing or disturbing the data mining operation. (Chapter 5 addresses these issues in more detail.)

Scalability. In a large-scale setting, algorithms must be designed to be resource-efficient (*e.g.* in terms of bandwidth, computation, and latency) in order to support a high number of parties contributing to the data analysis operation. Privacy and other requirements (*e.g.* reusability) of a data analysis application introduce various architectural design models to support scalable computation. The first contributions in direction of scalable (large-scale) secure computation were made by Saia and Zamani [381] and Boyle *et al.* [73], who introduced concepts such as communication locality[20]

[20] Measure to determine the number of communication partners in the protocol, per party.

and load balancing[21] to observe scalable behaviour. Nevertheless, currently deployed algorithms for data analysis and data release are not necessarily capable of being deployed in the context of large-scale computation and are not comparable to the performance of processing on plain data. These protocols have various shortcomings, for example: communication overhead [227, 139], impracticable parameter sizes [121, 190] and the need for expensive cryptographic operations [286] (in the presence of malicious adversaries). Therefore, achieving scalable and privacy-preserving computation (which becomes increasingly relevant) remains an open research question.

Efficiency. Similar to the scalability problem of data analysis algorithms, the trade-off between efficiency and privacy must remain at a reasonable level to be considered for real world applications by software developers and system architects. While in the early days of theoretical research only the feasibility of such algorithms was considered, researchers nowadays are slowly aiming to provide practicable solutions to be applied to real world applications [12]. In particular, algorithms have to be optimised to reduce the resource costs such as communication costs (number and size of each message sent between the parties), computational costs, and latency (number of communication rounds) in order to be applicable a wide variety of applications including lightweight and resource limited devices (*e.g.* in the Internet of Things (IoT)).

Utility (Accuracy). The discovered knowledge of the outcome of a data analysis task should accurately portray the contents of the analysed data [8]; moreover, the results should be useful with respect to the underlying application. Any imperfectness should be represented in terms of uncertainty — quantifiably, to retain measurable and arguable comparison and differences. Any added noise or exceptional data points to preserve privacy should be handled elegantly. When optimising the data analysis task for efficiency and scalability, the designer must be aware of the balance between data utility (for example, accuracy) and privacy to not diminish the advantage gained from the potential to analyse data. We come back to a more detailed discussion of the privacy/utility trade-off in Section 2.10, where we present measures to quantitatively evaluate such a trade-off.

Flexibility. Due to the complexity and interdisciplinary of privacy, over the years, many (privacy-preserving) data analysis algorithms have been introduced. Yet, many of those algorithms tackle similar tasks, operate in a similar fashion or build upon the same underlying principles. Nevertheless, until now, there exists no generalised framework covering (privacy-preserving) operations for these tasks. It remains an open research question to establish standardised, well-analysed, reusable and flexible algorithms that can be used (and reused) to cover various closely related data analysis operations.

[21] Measure to determine the distribution of resource costs in the protocol, per party.

2.7 Privacy-Preserving Data Analysis and Data Release

The concerns regarding the preservation of an individual's privacy in data release and data analysis has led to the research area of privacy-preserving data release and data analysis. In this section, we identify the two-fold considerations of privacy within this approach. Next, we state the trust assumptions within this model. Finally, we elaborate and review the current practice and techniques in this area and refer to subsequent sections and chapters where these techniques are discussed in more detail.

2.7.1 Privacy Considerations

Protecting an individual's privacy is typically a two-fold process. In the first step, raw sensitive information such as names, gender, addresses and so on should be modified or removed from the original data set in order to retain *individual privacy*. In the second step, sensitive knowledge, which can be inferred from applying data analysis techniques on the data set, must also be excluded, since this knowledge can equally well be used to compromise an individual's privacy. We refer to such privacy considerations as protecting *collective privacy*. Thus, we (informally) define:

Definition 7 (Privacy-Preserving Data Release and Data Analysis)**.** *Privacy-preserving data release or data analysis refer to tasks which enable a party to perform any collaborative operation on aggregated data [9] pertaining to multiple record owners, without diminishing the privacy of*

(a) an individual, and

(b) an individual within a collection of other individuals data.

As such, the primary goal of such analysis is to protect an individual's privacy (*i.e.* personally identifiable information (PII) [8] that can be linked, directly or indirectly, to an individual person; see Section 2.3). Protecting solely personally identifying data may not be enough. It is important, in many cases, to limit the amount of sensitive information which is revealed by the data analysis task[22] itself. In this context, collective privacy is concerned to limit the discovery of some patterns or trends which shall not be discovered in aggregated data.

2.7.2 Trust Assumptions

Dependent on the data release or data analysis operation and/or the overall application, various different architectural approaches (as discussed by Laud *et al.* [278]) can be exploited in order to preserve the privacy of a party. Complementary to the architectural settings come different trust assumptions of the participating parties within these settings. Broadly, we distinguish between an untrusted and trusted data curator model.

[22] These restrictions are typically context dependent and should not be taken as a fixed requirement.

In the former setting, record owners may not trust anyone with their personal information. As such, data remains distributed among various parties. In order to perform the required data analysis operation, intermediary values (related to the original values) are shared between the participating parties. These types of techniques are typically based on cryptographic approaches, in the literature known as *secure multi-party computation* (MPC). Protocols in the MPC domain can be broadly classified into two categories: those secure against semi-honest adversaries and those secure against malicious adversaries. While in the semi-honest setting, an adversary strictly follows the MPC protocol and tries to infer knowledge from the shared intermediate values, these restrictions do not apply for malicious adversaries. Therefore, a malicious adversary may arbitrarily deviate from the protocol and block, modify or replay any message or synthesise falsified messages. We discuss this model and its implications to privacy for any operational task in more detail in Chapter 5.

In the latter setting, the trusted curator model, we assume the existence of a trusted third party (TTP) [8] — a party that must be trusted by every participant to perform the beforehand agreed operations (*e.g.* data analysis tasks). Such a trusted party often works as an intermediary, in the form of a single and centralised party, between the parties participating in the data analysis operation. Every party willing to contribute to the data analysis operation must therefore be confident in sharing their data with the TTP, which will perform the data analysis operation on the union of the aggregated data and return or publish the result. Privacy-enhancing/preserving operations can be deployed on either the client side (*i.e.* at the parties, before sharing their data) or the TTP, or on both. This decision must be made during the design phase of an application and can't be taken lightly. Performing privacy-preserving operations on the client side will typically diminish the data utility on the TTP side, but enhance the level of privacy for each party. If such operations are waived, data utility on the TTP might be increased, but the TTP must be blindly trusted not to leak any information.

In the trusted third party model, we may further distinguish between an untrusted and trusted data analyst. In general, in this model, data analysts are considered to be untrusted. This is only natural, since the data analyst is the party which typically queries the privacy-preserving system and learns information about the underlying data set from the responses. Moreover, in the case of a trusted data analyst, we wouldn't need to worry about privacy issues from any data release or data analysis. Thus, a privacy-preserving system needs to limit the amount of information a data analyst is able to infer about the underlying data set or restrict access after a pre-defined number of queries to the system.

2.7.3 Current Practice and Techniques

Modern considerations of privacy in data analysis gained momentum when, in 2000, two contributions titled *Privacy-Preserving Data Mining* were published. Earlier work with regards to privacy in data analysis and data release include the seminal work of Dorothy Denning [146, 145, 144] on statistical databases in the 1970s and David Chaum [102] on blind signatures in the 1980s. In [24], Agrawal and Srikant present general methods to hide sensitive information in datasets — by distorting the data val-

ues while preserving the underlying distribution. In [289], Lindell and Pinkas show the application of cryptographic tools (*e.g.* oblivious transfer) to preserve individuals' privacy for data mining operations (in their case, the ID3 classification algorithm). In [285], the authors extend their work in this area. In [361], Pinkas shows the application of secure distributed computation techniques to demonstrate their relevance to the privacy-preserving computation of data analysis algorithms. Moreover, Lindell and Pinkas [290] survey basic paradigms and notions of secure multi-party computation and show the relationship and relevance to privacy-preserving data analysis.

In early research, the definition of privacy in privacy-preserving data analysis was somewhat vague. As such, in [117], Clifton *et al.* provide the reader with a framework and metrics for discussing privacy-preserving data mining as a foundation for further research. Agrawal *et al.* [23] discuss metrics for the design and quantification of privacy of privacy-preserving data analysis algorithms, thus illustrating the effectiveness of (in their case) perturbation-based algorithms. Besides, Clifton *et al.* [118] point to another (different) problem: there exist a lot of different solutions for various data mining operations; however, in many cases, the underlying principles and operations are similar (or closely related). In response to this problem, they introduce a tool set of basic privacy-preserving operations that can be assembled and applied to various data mining applications.

One of the first books [423] introducing and summarising the extensive research developments to that date was by Vaidya *et al.*, who point out techniques according to the common classes of data analysis. In [421], Vaidya and Clifton present a general introduction to, and overview of, techniques for privacy-preserving data analysis.

In addition, a fairly extensive overview of techniques, models, considerations and applications of privacy-preserving data analysis is given in [22]. (In this context, Aggarwal and Yu [19] survey and review the state-of-the-art methods for privacy-preserving data mining models and algorithms; Kantarcioglu [257] surveys techniques for horizontally-partitioned data sets, while Vaidya [422] surveys techniques for vertically-partitioned data sets.) More recent surveys in this area are [21, 156, 300, 20, 450, 316]. More specifically focused on the area of privacy-preserving data release is the survey of Fung *et al.* [187], which presents a systematic overview of recent techniques in this area.

Importantly, Bertino *et al.* [60] present a framework to evaluate privacy-preserving data analysis algorithms. In particular they evaluate which of the techniques better preserve sensitive information, the quality of the data resulting from the privacy modifications, and the performance of the algorithms.

With the introduction of differential privacy [159] in 2006 there has been a shift [119] from syntactic privacy models, and, more generally, from models of non-interactive (privacy-preserving) data release to interactive (privacy-preserving) data analysis and the use of probabilistic privacy models. This shift is mainly due to the inevitable problem of auxiliary information within the mode of non-interactive data release. Nevertheless, in some application scenarios it is still required to release data sets; as such, it is necessary to provide 'reasonable' privacy mechanisms for these cases.

In the following sections and chapters we go into more detail about the models, techniques, metrics and applications used in privacy-preserving data release and data analysis. Over the years, three major strands of research have evolved: (a) syntactic privacy models, (b) probabilistic privacy models (mainly, differential privacy), and (c) cryptographic approaches (based on secure multi-party computation). We introduce the current practice regarding the techniques of strands (a)-(c) in Section 2.8; we explore our research on strand (a) in Chapter 3, on strand (b) in the following sections, and on strand (c) in Chapter 5. The models enforce a certain 'structure' or outline requirements to which the original data need to be transformed in order to fulfill the privacy requirements that those models guarantee. Operations, mechanisms and techniques to ensure such a structure are presented in Section 2.9. Furthermore, it is important to evaluate privacy mechanisms and models regarding the ability to preserve privacy. As such, in Section 2.10, we present metrics to measure the privacy/utility trade-off, data utility, efficiency and performance of a privacy mechanism.

2.8 Privacy Models

2.8.1 Definition of a 'Privacy Model'

Privacy models set out the 'structure' or requirements to which plain data values have to be transformed in order to achieve the guarantees (*e.g.* preserve an individual's privacy, preserve collective privacy) specified by the same. As such, a privacy model comprises a statement of a privacy guarantee it aims to convey (*e.g.* all entries in a published data set are indistinguishable from k other entries) and a set of requirements, constraints or guidelines on the structure of how release data must look. In other words, the 'released data' complying to a privacy model is in a well defined format. Plain data, which is then transformed into this format, is assumed to provide privacy, with respect to the defined privacy assumption. Privacy-preserving mechanisms or operations (Section 2.9) are used for the transformation operations.

Definition 8 (Privacy Model)**.** *A privacy model sets out the structure or requirements to which released data values have to be transformed. As such, the privacy model defines a set of rules pertaining to how the data must look.*

Privacy models differ in the levels of privacy-preservation and can be broadly classified into categories based on their attack principles. In particular, this includes privacy-preservation for table linkage, entry (or record) linkage and attribute linkage. The former defend against linking an individual to the dataset itself, to an entry in the published dataset, or to a sensitive attribute in this dataset, respectively. There may exist other models protecting against a variety of privacy issues, though, here, we focus on the most relevant and typical problems. Next, we look at two different categories into which we may assign privacy models.

2.8.2 Syntactic vs Probabilistic Privacy Models

Following the developments in current literature [119], privacy models may be distinguished to follow either a *syntactic approach* or a *probabilistic approach*. A syntactic privacy model imposes a given structure on the data (*e.g.* indistinguishable from k other entries, contains ℓ types of different 'sensitive' elements in a release group, distribution of elements within release groups and the overall data set is close to a value t). As such, privacy is ensured through a 'syntactic' definition of the data. Accordingly, the 'syntactic' definition of the data must match what the privacy guarantee aims to convey. In other words, the syntax defines the privacy (*e.g.* syntax 'k-indistinguishable values within a group' map to notion 'indistinguishability'). In this context, due to the reliance of structure in the definition of privacy, such privacy models normally do not come with formal treatments or properties (*e.g.* composability, group privacy, post-processing privacy) which allow to analyse the privacy model in a broader sense. Syntactic privacy models typically follow a (non-interactive) data release mode. Due to their syntactic structure, the result of two queries asking the same question will always be the same.

A probabilistic privacy model is based on uncertainty at an individual level. Uncertainty may be achieved through the injection of noise or other sources of randomisation used in a privacy mechanism. In this context, these models are typically defined on the difference between an adversary's prior beliefs of some event or knowledge about data and her posterior beliefs of observing a response to a query (of a system with an underlying probabilistic privacy model). Consequently, privacy is defined on the difference between an adversary's prior and posterior beliefs and not on the structure of the released data. Such a relaxation of constraints allows one to argue about privacy in less limited settings (*e.g.* an adversary may be allowed to learn all but one entry, there is no limitation through existing auxiliary information, released data must not be compliant to certain structure). What is more, probabilistic privacy models (*e.g.* differential privacy) are supported by formal proofs and allows one to reason about privacy in a strict mathematical setting. These kind of models typically follow a (interactive) data analysis mode. Due to the randomisation, two identical queries to such a system can result in different answers.

2.8.3 Privacy Axioms

Contrary to the goals of syntactic models, for modern (probabilistic) privacy models it is argued that the 'privacy-preserving' behaviour should be inherent to the algorithm which is complying with the guidelines of the privacy model. In other words, 'to preserve privacy' is a property of an algorithm and should not be imposed, for example, only on the output of the same. Thus, to ensure internal consistency of privacy definitions two privacy axioms have been proposed by Kifer and Lin [263]: *transformation invariance* and *convexity*. (In the context of Axiom 1 and 2, Kifer and Lin make use of the terms randomised algorithm, privacy mechanism and privacy definition. These definitions may vary from ours. Thus, for details we refer to the original paper [263]. Briefly stated, a randomised algorithm is a conditional probability distribution of an

input value to a set of output values; a privacy definition is a set of randomised algorithms we trust to perform a transition from plain data to sanitised data; and a privacy mechanism is a randomised algorithm which satisfies the privacy definition.)

Axiom 1 (Transformation Invariance [263]). *Let $\mathcal{M}$ be a privacy mechanism for a particular privacy definition and let $\mathcal{A}$ be a randomised algorithm whose input space contains the output space of $\mathcal{M}$ and whose randomness is independent of both the data and the randomness in $\mathcal{M}$. Then $\mathcal{M}' = \mathcal{A} \circ \mathcal{M}$ must also be a privacy mechanism satisfying that privacy definition.*

Axiom 1 deals with any post-processing of sanitised data. Informally it states that any post-processing via a randomised algorithm $\mathcal{A}$ of the results of a privacy mechanism $\mathcal{M}$ should satisfy the same privacy definition as if this post-processing hasn't taken place. This property follows naturally from the idea that if a privacy mechanism reduces the amount of knowledge contained in the released data of said mechanism any post-processing without access to the sensitive data won't add new information that an adversary is able to leverage in order to break an individual's privacy. (We note that, under the definition of Kifer and Lin [263], the identity function is a privacy mechanism. Consider $\mathcal{M}$ to be an encryption algorithm. Then, $\mathcal{A}$ may be its inverse — the decryption algorithm. It follows that $\mathcal{A} \circ \mathcal{M}$ is the identify function; *i.e.* $\mathcal{M}'(d) = d$ for any data d.)

Axiom 2 (Convexity [263]). *Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be privacy mechanisms that satisfy a particular privacy definition (and such that the randomness in $\mathcal{M}_1$ is independent of the randomness in $\mathcal{M}_2$). For any $p \in [0, 1]$, let $\mathcal{M}_p$ be a randomized algorithm that on input i outputs $\mathcal{M}_1(i)$ with probability p (independent of the data and the randomness in $\mathcal{M}_1$ and $\mathcal{M}_2$) and outputs $\mathcal{M}_2(i)$ with probability $1 - p$. Then $\mathcal{M}_p$ is a privacy mechanism that satisfies the privacy definition.*

Axiom 2 deals with the effects of selecting a privacy mechanism at random. This means, considering both mechanisms $\mathcal{M}_1$ and $\mathcal{M}_2$ satisfy a privacy definition, the selection of either one of them should add enough uncertainty to the sanitised outcome that one is not able to infer about the input data. As such, we should be free to choose whichever mechanism we like as long as it is not dependent on the real input values. Privacy mechanisms which satisfy this axiom support flexible interchangability of themselves.

2.8.4 Three Popular Strands of Privacy Models: Syntactic, Probabilistic and Cryptographic

Current privacy models can be attributed to one of the following three major strands: (a) syntactic privacy models, (b) probabilistic privacy models (mainly, differential privacy), and (c) cryptographic approaches (based on secure multi-party computation). In the following, we reference privacy models of each approach and briefly outline them. For a more detailed introduction, we refer the reader to the following chapter(s), a survey by Fung *et al.* [187], or the original papers (as referenced).

Strand (a): Syntactic Privacy Models. In the absence of clear definitions of notions for privacy and their formal treatment in a precise mathematical framework,

practitioners have been templed to apply various rule-of-thumb mechanisms to protect privacy. For example, "reject query answers that include fewer than x entries", or "the revealed results must include k-indistinguishable entries and pertain to certain value distributions within those entries".

Prominent models of the syntactic approach are:

(a) **k-anonymity.** Introduced by Samarati and Sweeney [384, 408], it guarantees that each individual's data in a sanitised dataset is indistinguishable from the data of at least $k-1$ other individuals[23], whose data is within the original dataset. Thus, each quasi-identifying tuple in the sanitised dataset occurs at least k times. The sanitised dataset is then called k-anonymous. k-anonymity may be achieved by supression or generalisation of original data values.

(b) **ℓ-diversity.** Introduced by Machanavajjhala *et al.* [298] to prevent homogeneity [298] and background knowledge [298] attacks[24], which the k-anonymity model is susceptible to. ℓ-diversity extends the k-anonymity model: a sanitised dataset is ℓ-diverse if every group of k-anonymous quasi-identifying values map to ℓ well-represented sensitive attributes.

(c) **t-closeness.** Introduced by Li *et al.* [282] to prevent skewness [282] and similarity [282] attacks[24], which the ℓ-diversity model is susceptible to. t-closeness extends the ℓ-diversity model: a sanitised dataset is t-close if the distance between the distribution of a sensitive value within a group is not more than a threshold t compared to the distribution of the sensitive value in the whole dataset.

Other syntactic privacy models include: β-likeness [97], $(1,k)-$, $(k,k)-$, and global $(1,k)$-anonymisation [193], k-concealment [409], p-sensitive k-anonymity [417], (X,Y)-privacy [431], (LKC)-privacy [186], k^m-anonymity [410], ρ-uncertainty [98], $k^{\tau,\epsilon}$-anonymity [201], δ-presence [330], c-confident δ-presence [331], m-invariance [443], multirelational k-anonymity [332], (α,k)-anonymity [440], (k,e)-anonymity [452], personalized anonymity [442] and ff-anonymity [430].

Strand (b): Probabilistic Privacy Models. Privacy models which follow the probabilistic approach are informed by the 'uninformative principle' by Machanavajjhala *et al.* [298], which states the following:

Definition 9 (Uninformative Principle)**.** *A published (sanitised) database should provide the adversary with little additional information beyond her background knowledge. As such, the difference between the prior and posterior beliefs of the adversary should not be large.*

The prior beliefs of an adversary is any information she has obtained (*e.g.* about a certain individual, whose data may be contained in the database) prior to interacting with the sanitised database. The posterior beliefs is the information she holds after interacting with the sanitised database. Therefore, the 'uninformative principle' is defined on the difference between prior and posterior beliefs of an adversary — a

[23] It is assumed, that each entry in a dataset belongs to exactly one individual.
[24] For a more detailed discussion of these attacks, we refer the reader to Section 6.4.

probabilistic privacy models is successful in preserving an individual's privacy if the change in an adversary's belief is small[25].

Prominent models of the probabilistic approach are:

(a) **Differential privacy** and its variants. The techniques for the notion of differential privacy emerged from a series of papers [64, 151, 162] and was formalised by Dwork *et al.* [159]; various authors [161, 319, 132, 297, 88, 158, 292, 150, 404, 154, 218, 370, 318, 165, 25] extended the pure notion of ϵ-differential privacy. As an abstraction, the notion of differential privacy guarantees that the difference of a 'view' of a dataset *including* the data of an individual changes only negligibly compared to a dataset which does not include the data of the same individual. In other words, both views should be indistinguishable. In its simplest form, the privacy level is controlled by a real number $\epsilon > 0$, which determines the privacy loss of individuals. From the viewpoint of an analyst, ϵ regulates the knowledge gain of the analyst.

Other probabilistic privacy models include: (c, t)-isolation [104], (d, γ)-privacy [371], (ρ_1, ρ_2)-privacy [179] and distributional privacy [65].

Strand (c): The Cryptographic Approach. Various cryptographic techniques allow the privacy-preserving analysis and release of data. In the cryptographic setting, data is typically released to a set of authorised participants. As such, these techniques are concerned with preserving computational privacy. Release privacy may be ruled out-of-scope in such a setting due to the release to the set of authorised participants.

Prominent techniques of the cryptographic approach are:

(a) **Secure multi-party computation (MPC)** allows a set of parties, $P_1, \ldots, P_n$, each with their private input, $x_1, \ldots, x_n$, to jointly compute a function $f(x_1, \ldots, x_n) = y$. The collaborative function f can be any function, and may be a different function for each of the participating parties. The parties are distrustful to each other and don't want to reveal their private inputs to the other parties. As such, an algorithm is said to be secure in the multi-party setting [290] if it fulfills the properties of *privacy*, *correctness*, *independence of inputs*, *fairness*, and *guaranteed output delivery*. Such an algorithm can achieve different levels of security, depending on the behaviour of a static or an adaptive adversary, namely semi-honest (or 'passive'), covert or malicious (or 'active') security. Later, in Section 2.11, we introduce this approach in more detail.

(b) **Homomorphic encryption** ensures that operations performed on the ciphertext map to the same or a related operation performed on the plaintext. In other words, if an operation is performed on the two ciphertexts, for example an addition, then the resulting ciphertext, when decrypted, equals the operation result as if it would be performed on the plaintexts directly. Acar *et al.* [14] provide an overview of various homomorphic encryption schemes.

[25] The definition of 'small' is specific to the probabilistic privacy model. For example, in differential privacy the privacy budget ϵ regulates the privacy/utility tradeoff — as such, the amount of information an analyst is allowed to learn about the dataset.

The two above mentioned techniques can be directly applied in a distributed multi-party setting to ensure computational privacy. There exist many other techniques which can be used to ensure computational privacy — those techniques may be more subtle, require 'integration' or combination with other approaches to be directly applicable to ensure privacy-preserving data analysis. Typically, these techniques are designed for 'different' objectives than privacy-preserving data analysis, but are still used in the same (as 'building blocks' to guarantee the properties set out by the techniques above). For example, these techniques include: oblivious transfer [365], secret sharing [152] and zero-knowledge proofs [258].

2.9 Operations, Mechanisms, and Algorithms to achieve Privacy

We categorise privacy-preserving 'transformations' according to their abstraction level into three groups: algorithms, mechanisms and operations. The different abstraction levels can be understood and 'ranked'[26] as follows:

- **algorithm:** at the lowest level of abstraction, an algorithm outlines detailed steps of the transformation (*i.e.* step-by-step instructions from a pre-condition to a post-condition of the data)
- **mechanism:** at the mid-level of abstraction, a mechanism abstracts the mathematics (*e.g.* input distribution to output distribution of the data)
- **operation:** at the highest level of abstraction, an operation abstracts the transformation via a high level language (*i.e.* abstract description how the data is to be transformed; described, for example, using natural language)

2.9.1 Privacy-Preserving Operations

Privacy-preserving *operations* are the most basic building blocks — an 'operation' sets out how to transform the private data to be included in the public release. The operation is a high level abstraction of a transformation.

Definition 10 (Privacy-Preserving Operations)**.** *A privacy-preserving operation is a high level description of a privacy transformation. An operation abstracts the 'idea' of how to manipulate raw sensitive input data into a output stream which follows the rules set out by the privacy model.*

A privacy-preserving operation may follow the rules of a privacy model or may be completely independent and standalone. Typically, in a privacy model multiple privacy operations work together to satisfy the rules set out by the privacy model.

Privacy-preserving operations either follow the *aggregation* approach or the *randomisation* approach. In the aggregation approach, several singular values are combined

[26] This ranking may be understood as the abstraction from human readable natural language down to machine readable byte code. In essence, non-experts may be more familiar to understand high level abstractions such as natural language as opposed to the complexities that arise from low level codes.

Table 2: Summary of privacy-preserving operations for data analysis.

Type	Principle	Privacy Level‡	Data Utility‡
aggregation	aggr. based	med. - high	high
anatomisation	aggr. based	high	high
permutation	aggr. based	low - high†	high
generalisation	rand. based	low - med.	low
suppression	rand. based	high	low
perturbation	rand. based	med. - high	med.
replacement	rand. based	high	low

† strictly dependent on the size of the dataset.

‡ strictly context dependent — the stated evaluation is from a general viewpoint, but might deviate widely in different settings.

and the anonymised (privacy-preserving) result is released only if a certain amount of non-distinguishable elements exist, or if the linking relation between the data entry and an individual is broken. (A summary of these operations is given in Table 2[27]. Additionally, a continuous example illustrating the effects of these privacy operations is given in Appendix A.)

(a) **Aggregation** [9]. In this technique, data values of individuals are aggregated either in a *spatial* or in a *timely* manner. The data analysis operation is applied directly on the aggregated values. The data utility of this approach remains unchanged, since the original values are not modified. Depending on the application, the level of achieved privacy varies between medium to high.

(b) **Anatomisation** [9]. In this technique, the linking relationship between the data values and the individual gets disassociated. The data utility for this approach remains unchanged, since the original values are not modified. However, using this approach it is not possible to post hoc determine the associated original value from the anonymised value. The privacy level of this approach is high.

(c) **Permutation** [9]. In this technique, the data entries normally associated to an individual are shuffled randomly to break the association to that particular individual. Similar to the other mentioned approaches, the data utility remains unchanged. Depending on the aggregation size, the level of privacy varies between low to high.

[27] In Table 2 we evaluate the privacy operations with regards to their privacy level and data utility into three categories: low, med., and high. The author recognises that giving a strict definition of these categories may not abstract all applications. In this context, this categorisation should be seen as a qualified 'opinion' of the author and not as a strictly unchangable definition. The rationale behind our 'definitions' is as follows: for data utility, the aggr. based methods retain high utility since the data elements remain unchanged; generalisation, suppression and replacement retain low utility since the data elements are typically changed to a high degree; perturbation typically slightly modifies the original values. For privacy, aggregation and permutation is highly dependent on the dataset size, thus the privacy levels vary; anatomisation de-associates the sensitive attributes from the personally identifying attributes, suppression and replacement substitutes the original attributes with replacement values, thus achieving high privacy guarantees; generalisation generalises to a related element, where the privacy level depends on the granularity of the taxonomy; and the privacy level of perturbation remains typically highly dependent on the amount of noise added to the original values.

The other category of ensuring privacy-preserving processing is the randomsation approach. In this approach, data values get obscured by the addition of random noise from a uniformly random distribution either directly to the input values (on the client side) or to the output of the data analysis result. These techniques keep the linking relation between the data value and the associated individual.

(a) **Generalisation** [9]. In this technique, attributes are transformed from a specific level to a more general level via a pre-defined taxonomy. For example, attributes containing alphabetical characters are raised via a taxonomy, while numerical attributes are replaced by an interval. Dependent on the range of the taxonomy/interval, the data utility is diminished. The level of privacy ranges from low to medium, depending on the sensitivity of the attribute.

(b) **Suppression** [9]. In this technique, sensitive attributes are replaced with a placeholder element, such as an asterisk, *. While this diminishes all data utility, it would achieve perfect privacy. This technique should only be used rarely, when there are only a few distinctive attributes left, which can't be preserved otherwise.

(c) **Perturbation** [9]. In this technique, a random noise value selected, for example, from a Laplace or a uniform distribution, is added to the sensitive attribute or to the data analysis result. The resulting data utility is diminished slightly, while the privacy level is guaranteed to be medium to high.

(d) **Replacement** [9]. In this technique, the original values are replaced with replacement values, where the new distribution of the values is kept similar to the original distribution. In this sense, the data analysis result remains valuable and the original values are hidden. Applying this technique, the data utility of the singular values is completely diminished, while keeping the overall distribution intact. On the other hand, this technique attains perfect privacy protection (*i.e.* nothing about the original values is revealed).

2.9.2 Privacy Mechanisms

A privacy mechanism gives a mathematical abstraction of the privacy transformation. For example, a privacy mechanism describes how an input distribution of data should be transformed to a privacy-preserving output transformation.

Definition 11 (Privacy Mechanism)**.** *A privacy mechanism expresses the mathematical abstraction which transforms sensitive input data to public 'privacy-preserved' data. The mechanism is associated with a privacy model; therefore, the mechanism is informed by the rules and structure set out by the privacy model.*

Diaz *et al.* [149] refer to a privacy mechanism as probabilistic mappings that satisfy a privacy definition. Further, they present privacy mechanisms for local differential privacy. Ghosh *et al.* [192] present universally utility maximising privacy mechanisms while satisfying the privacy requirements of differential privacy. Pai and Roth [344] survey recent work at the intersection of mechanism design and privacy; their main

focus point is mechanisms that satisfy the privacy requirements of differential privacy. Other examples of privacy mechanisms (again, with a focus on differential privacy) are the Laplace mechanism [162], the exponential mechanism [314] and the median mechanism [378].

2.9.3 Privacy Algorithms

A privacy algorithm expresses the steps necessary to transform sensitive data to public data. In this context, a privacy algorithm fulfills the rules set out by a privacy model. Further, a privacy algorithm implements one or more privacy operations and typically follows a privacy mechanism.

Definition 12 (Privacy Algorithm)**.** *A privacy algorithm is a list of instructions which transforms sensitive input data to public 'privacy-preserved' data. The algorithm implements a privacy operation and follows a privacy mechanism. It is a detailed step-by-step list of 'how' to transform the well-defined input data to well-defined output data.*

Fung *et al.* [187] survey anonymisation algorithms which comprise algorithms to prevent record linkage, attribute linkage and table linkage attacks as well as probabilistic attacks. Aggarwal and Yu [19] present privacy algorithms for data analysis focusing on the previously discussed privacy operations. Dwork [160] surveys algorithms satisfying differential privacy; further, Dwork and Roth present algorithmic foundations for differential privacy in [163].

2.10 Information Metrics: Measuring the Privacy/Utility Trade-Off

In the design of any privacy mechanism a variety of factors to optimise for efficiency, utility or privacy need to be taken into account. For example, domain size, scale and shape of the data, number and distribution of datasets, number and type of queries to the collective data, measuring errors such as sampling errors and errors induced by the privacy mechanism, data-dependent and data-independent optimisations and pre/post processing procedures to name a few.

These factors additionally have a significant impact on the data utility. While one of the goals of a privacy-preserving transformation is to reduce/obfuscate the sensitive information, the resulting data still needs to be useful enough for further processing. In other words, the information contained in the public releases must be useful enough to extract representative data points.

2.10.1 Privacy Metrics

Wagner and Eckhoff [427] survey multiple privacy metrics from various technical domains including communication systems, databases, smart-metering and genome privacy. Despite their diversity, the authors classify the privacy metrics into four common

characteristics: adversarial goals, adversarial capabilities, data sources, and input and output measures. While there exist many possible categorisations of privacy metrics the most intuitive one is with regards to output metrics. Output metrics analyse the utility/privacy tradeoff from the output of a privacy mechanism. It is important to recognise that there is no single metric to cover all different aspects of privacy. Wagner and Eckhoff [427] present a taxonomy of eight output metrics: (a) uncertainty, (b) information gain or loss, (c) data similarity, (d) indistinguishability, (e) adversaries success probability, (f) error, (g) time and (h) accuracy or precision. For each category, Wagner and Eckhoff list specific metrics. We refer the reader to their excellent survey [427] for details.

2.10.2 Data Metrics and Search Metrics

Information metrics are used to measure accuracy and data usefulness. Moreover, those metrics may determine the effectiveness of anonymisation operations. Fung *et al.* [187] survey various information metrics. In their survey, they distinguish between data metrics and search metrics.

Definition 13 (Data Metric)**.** *A data metric evaluates the data quality of the anonymised dataset with respect to the data quality in the original dataset.*

Definition 14 (Search Metric)**.** *A search metric evaluates the data quality by steering a search algorithm to identify the maximum information gain or minimum distortion of a released anonmyised dataset.*

A search metric achieves the best outcome by ranking a set of possible anonymisation operations and then 'greedily' performing the best step at each point in the search. Both types of metrics share common similarities in evaluating the data utility of the release anonymised dataset.

2.10.3 General Purpose, Special Purpose and Trade-off Metrics

Fung *et al.* [187] further distinguish privacy metrics with regards to their applicability and assumptions of their use.

General Purpose Metrics. In many cases, a data publisher does not know how the data may be further used or analysed; an information metric for one data analyst may be completely useless to any other. General purpose metrics are trimmed to operate optimal given such environments. An example is to use metrics of type (c) data similarity [427] that measure the 'similarity' of the plain data and the anonymised data. These kind of metrics typically 'charge' a penalty for each instance of a value that is generalised or suppressed. To illustrate, an example would be to generalise a value *Engineer* according to a given taxonomy to its more general value *Professional*. Depending on the data metric, a different penalty is given. Examples of general purpose metrics are: minimal distortion metric [383], ILoss [442], discernibility metric [398] and distinctive attribute [407].

Special Purpose Metrics. If the purpose of the data analysis is known at the time of publication of the anonymised data, the anonymisation algorithm can be chosen fittingly or optimised to retain better results. The goal here is to generalise or suppress only attributes which do not influence the data analysis operation. The data is still published in an anonymised way allowing the data recipient to run any data analysis operation[28]. In a classification example, future cases are classified to predefined classes according to training data. Thus, an anonymisation algorithm may generalise or suppress noisy data, but retain the classification information utilised by the classifier. For example, a person's birth year may be useful for the classifier, yet, the full birth date may be noise. An example of a special purpose metric (for classification) is the 'classification metric' [246]. The idea of this metric is again to charge a penalty for each suppressed or generalised record. As opposed to the general purpose method, where, for each invocation, a penalty is charged, here, the penalty counts if the record's class is not the majority class of its group.

Trade-off Metrics. While in a special-purpose metric the goal is to optimise the data usefulness for a given data analysis operation, the idea of trade-off metrics is to find an optimal balance between the two requirements. These metrics follow type (b) information gain or loss in the taxonomy of Wagner and Eckhoff [427]. An example of such a metric is described by Fung *et al.* [184, 185] based on the principle of information / privacy trade-off. The idea of the metric is to iteratively specialise a general value into its child nodes. For each step, the information gain and the privacy loss can be calculated and the search metric can be optimised accordingly.

2.10.4 Additional Anonymisation Metrics

Other metrics to measure the 'anonymity' of a data release are described by Serjantov and Danezis [394], Edman *et al.* [167], Diaz *et al.* [147, 148], Dwork [159], Clauß and Schiffner [116] and Andersson and Lundin [30]. In addition to the previously mentioned metrics, metrics from the field of information theory such as data entropy and distribution of data values can be applied to measure the information gain or data usefulness.

2.11 The Cryptographic Approach: Secure Multi-Party Computation

Previous approaches typically assumed the existence of a trusted third party (TTP) — a party that must be trusted by every participant to perform the beforehand agreed

[28] In case the data operation is known in advance, the data holder may run the data operation itself and publish only the result. This, however, is not practical for the non-expert data holder nor desirable, since it limits the data recipient to run any kind of data analysis operation. (In practice, the best data analysis result is gained by running and comparing multiple data analysis algorithms. Such processing retains the greatest flexibility for the data recipient.)

operations. Every participant in the communication protocol willing to contribute to the data analysis operation must be confident in sharing their data with the TTP.

A TTP is a strong assumption, which typically does not hold for applications in real-world settings. From this limitation, a number of techniques that are based on cryptographic approaches emerged, in the literature known as *secure multi-party computation* (MPC). Similarly to previous techniques, the new models enable multiple parties to collaboratively evaluate a data analysis operation on the union of their data — without revealing their sensitive values. These cryptographic approaches address the problem of *input* privacy. The major difference between those models (and the previous ones) is that the data of each participant (*i.e.* the datasets) is not shared directly with the other parties and remains distributed over all parties. In order to perform the data analysis operation, intermediary values (related to the original values) are shared between the participating parties.

The concept of 'secure multi-party computation' was introduced in the 1970s under the term of 'mental poker'[29] [396] predominantly by the works of Yao [447] (Millionaire's problem), Blum [66] (coin flipping over a distance) and Rabin [365] (oblivious transfer).

More recently, Lindell and Pinkas [290] survey basic notions and paradigms of secure multi-party computation with relation to privacy-preserving data analysis and data release. Other literature providing an overview of MPC include [27, 227, 341, 139, 142, 296, 94, 155] and publications of the PRACTICE[30] project (*e.g.* [342, 216]).

2.11.1 Definitions

Secure multi-party computation permits a collection of mutually distrustful parties to perform an evaluation of a collaborative function without any of the involved parties or computing servers learning anything about the inputs of other participating parties, except what can be determined from the output of the computation. More formally, the MPC problem is defined as follows:

Definition 15 (Secure Multi-party Computation)**.** *A set of parties* $P_1, \ldots, P_n$*, each with their private input* $x_1, \ldots, x_n$*, jointly compute a function* $y = f(x_1, \ldots, x_n)$*. Moreover, the evaluation of* f *must not reveal any knowledge about the private inputs* $x_1, \ldots, x_n$*.*

The collaborative function f is not restricted and can be any function, even different for all of the participating parties. An algorithm is said to be secure in the multi-party setting if it fulfills the properties of (a) *privacy*, (b) *correctness*, (c) *independence of inputs*, (d) *fairness*, and (e) *guaranteed output delivery* as defined in [290].

Protocols in the MPC domain can be broadly classified into three categories — those secure against semi-honest adversaries; against covert adversaries; and against malicious adversaries. These different levels of security can be further classified depending on the behaviour into static or adaptive adversaries.

A semi-honest adversary does not deviate from the given protocol, however she tries to gain knowledge about the other parties' inputs by participating in the protocol

[29] The term 'mental poker' was introduced to describe a set of cryptographic problems that are concerned with playing a fair game over a distance without the need of a trusted third party.

[30] https://practice-project.eu/

execution. A subtle more involved adversary type is that of a covert adversary, which only deviates from the protocol if it knows that it can't get caught.. The strongest type of adversary represents a malicious adversary, which arbitrarily deviates from the protocol to block, modify or replay any message or synthesise falsified messages.

2.11.2 Theoretical Research and Proof-of-Concept Protocols

The simplest solution to the MPC problem is to share the private data with a mutually trusted party — with that party being trusted to compute the function. However, a trusted relationship can not always be established and early results from the late 1970s/80s have shown the feasibility of replacing the trusted party with a cryptographic protocol to achieve the same level of security. In general, the TTP model is referred to as an *ideal world* model. The cryptographic approach to simulate such a protocol is the *real world* model. When proving the security of a MPC protocol (a.k.a. Real), the designer must show that his protocol is indistinguishable from the ideal world model (a.k.a. Ideal). In order to achieve the notion of indistinguishability the advantage of an adversary $\mathcal{A}$ is bound to a negligible value μ:

$$\mathbf{Adv}_{\mathcal{A}}^{MPC} = |Pr[\mathcal{A}^{Real} \Rightarrow \text{true}] - Pr[\mathcal{A}^{Ideal} \Rightarrow \text{true}]| < \mu$$

In recent years, secure multi-party computation has shifted from being merely a scientific research area towards being practical and deployable in real-world applications. It started in the late 1970s and 1980s from theoretical research and feasibility results of garbled circuits to achieve secure computation [7]. This includes the early works of Shamir [395] and Blakely [63] on secret sharing, the foundational work of Yao [447] on Boolean circuits and the subsequent evolving protocols GMW [195] and BMR [51]. Further, the BGW [57] protocol utilises arithmetic circuits. To achieve communication privacy between the participating parties, Yao and GMW utilises oblivious transfer [365], BMR and BGW make use of secret sharing [395]. In addition to the previously mentioned techniques, MPC protocols typically take advantage of various other techniques of modern cryptography such as homomorphic encryption schemes [130, 136], commitment schemes [129] and zero-knowledge proofs [136].

2.11.3 Practical Developments and Real-World Protocols

Due to the recent progress and significant improvements in computing technology, various proof-of-concept implementations and frameworks for secure multi-party computation have been implemented over the last two decades. These include the implementations of two-party protocols such as Fairplay [301], Tasty [226] and MightBeEvil [236], and implementation of multi-party protocols such as FairplayMP [56], Sharemind [67], VIFF [135], SEPIA [89], SPDZ [136] and SCAPI [169].

Table 3: Comparison of general frameworks for two-party protocols and multi-party protocols

Protocol	Type	Security	Cryptographic Primitive	Online/ Offline
Fairplay	two-party	malicious	Boolean circuits oblivious transfer	no
Tasty	two-party	semi-honest	garbled circuits†	no
			homomorphic enc.	
MightBeEvil	two-party	semi-honest	garbled circuits†	no
FairplayMP	multi-party	semi-honest	Boolean circuits	no
Sharemind	multi-party	semi-honest	secret sharing (add.)	no
VIFF	multi-party	malicious	arithmetic circuits	yes
			secret sharing (Shamir)	
SEPIA	multi-party	semi-honest	secret sharing (Shamir)	no
SPDZ	multi-party	malicious	arithmetic circuits	yes
			zero-knowledge	
SCAPI	multi-party	application dependent	commitments garbled circuits† oblivious transfer zero-knowledge	-

† arithmetic and Boolean circuits.

Table 3 shows a comparison between current[31] 'generic' two-party and multi-party protocols. Many of the evaluated protocols are based on garbled circuits and as indicated either secure in the semi-honest adversary model or in the malicious adversary model.

2.12 Trusted Computing

In the following, we present some preliminary information of Trusted Computing followed by some preliminary information of the Intel Software Guard eXtensions (SGX)[32].

2.12.1 Preliminaries on Trusted Computing

The following concepts — (a) to understand the concepts; (b) as basic building blocks; and (c) as enhancements to scalability and utility in the secure multi-party domain

[31] We acknowledge that the selected frameworks do not cover newest developments in the field of MPC research, and, consequently, might leave one with the mistaken impression that MPC is a limited approach or that it achieves only limited performance. However, our concern in this dissertation is to evaluate techniques that are capable of solving general MPC problems in large-scale and distributed environments. Thus, only a few MPC frameworks meet our requirements (some of which might be considered 'old') — and it is these that we base our comparison and later our benchmarking on. Further, the operations considered are due to restrictions imposed by what those frameworks can support. Our results should be seen through that perspective.

[32] https://software.intel.com/en-us/sgx

— are paramount for TRE systems. Trusted Computing (TC) is a complementary approach among others in the secure multi-party computation domain that introduces technologies and techniques that allow distrustful users to establish a trust relationship with each other or with a separate remote party. TC technologies are developed and promoted by an industry consortium — the Trusted Computing Group (TCG)[33], and include dedicated hardware and software components. Andrew Martin [303] presents a detailed overview of TC concepts. We refer further to [205, 204, 206, 207, 208, 210].

(a) **Trust.** 'Trust' in a relationship between two or more parties, is defined where one party, the *trustor*, is willing to rely on the statements and/or actions of another party, the *trustee(s)*. In this context, trust has the properties of being transitive between those parties, be it limited or unlimited over time.

In a computational sense, the TCG states 'trust' as follows:

Definition 16 (Trust [209])**.** *A party can be trusted if it always behaves in the expected manner for the intended purpose.*

Definition 16, however, does not exclude any bad behaviour. This means, if a system is designed to perform malicious operations it would be still be acceptable under Definition 16 as 'trust' establishment. To challenge such behaviour, the TCG expanded their definition (based on a definition of trust by Proudler [363]) as follows:

Definition 17 (Trust [363])**.** *A party can be trusted if it always behaves in the expected manner for the intended purpose; and the party can be unambiguously identified, operates unhindered and a user can forward experience about a consistent and good behaviour of the party.*

(b) **Trusted Computing Base.** The Trusted Computing Base (TCB) of a party comprises all hardware, firmware and software components that are critical to the security of the system. In other words, if there is a vulnerability in any of these components the security of the whole systems is jeopardized. Thus, it is recommended to keep the TCB of a party is as small as possible in order to minimise the costs for verification and to protect the confidentiality and integrity of its elements.

(c) **Software Attestation.** Software attestation [120] is a process to establish a trust relationship. In this process, a prover provides a statement to a verifier. The statement is typically a measurement, for example, computed by a cryptographic hash function over the code and data pages of the algorithm and operating system of the remote party. In more detail, during the launch of a party each code page loaded into the operating memory is 'measured' and stored into reserved registers or memory locations by extending the current value (*i.e.* computing a hash over the concatenation of the current value and the new page). These special registers then represent the current state of the remote party and can be exchanged for verification purposes.

[33] http://trustedcomputinggroup.org

To guarantee the validity of an attestation message, the verifying party must trust the *root of trust for measurement* [303] and the *root of trust for reporting* [303] of the proving party. The former element (or function) ensures the integrity of the measurement process (*e.g.* via a cryptographic hash function) and the latter element ensures integrity and validity of the reporting process (*e.g.* via a unique, verifiable platform identity). More generally, to ensure a valid attestation and successful trust establishment, the following properties should be fulfilled:

(a) **Authenticity.** The verifying party must be able to unambiguously determine whether the measurements were provided from the proving party.

(b) **Currentness.** The verifying party must be able to unambiguously determine whether the measurements represent the current state of the proving party.

To ensure authenticity, public-key signatures schemes (*e.g.* based on RSA[34] [204] and ECC[35] [210]) are typically used within attestation protocols. Using such 'primitive' systems may lead to privacy problems, for example, in case several verifier collude and track the behaviour of a prover. To overcome these privacy threats, for TPM v1.1 [204] the usage of a privacy certification authority (CA) was proposed, and for TPM v1.2 and v2 [206] the Direct Anonymous Attestation (DAA) protocol [81] based on RSA (for v1.2) and on ECC (for v2) was proposed.

To prevent the possibility to replay old attestation messages, timestamps or nonces may be applied.

Attestation may be performed locally (*i.e.* within the same system), which is referred to as *local* attestation, while attestation performed remotely (*i.e.* over the network to a remote party) is referred to as *remote* attestation. Performance of attestation protocols play an important role, concretely with remote attestation — a focus is given on latency (*e.g.* time of a single attestation) and scalability (*e.g.* rate of verifying parties in a give time interval). In this context, attestation protocols can be classified as: (a) one-to-one; (b) one-to-many; (c) many-to-many; or (d) many-to-one. The first is usually the case, where two parties attest to each other and establish a trust relationship. In a type (b) attestation protocol, the number of attestation messages over the network can be reduced by reusing the already performed measurement and sharing it to other parties — in return improving the scalability of the protocol.

Examples of attestation protocols are [204, 29, 81, 351]. (This is further discussed in Section 6.6.3.)

(d) **Secure Storage.** The memory space within the main processing unit (CPU) is typically very limited — thus, memory-expensive operations are required to swap out data and code pages to the next level memory location (*e.g.* LL1 cache, RAM, hard-disc). Moreover, operations that can not be processed instantaneously, rely on other processes, or are processed over a long time period, require temporary storage outside of the main processing unit. Storage space outside of the immediate processing unit is typically less secure, because other processor cores and

[34] The RSA (Rivest–Shamir–Adleman) crypto system [374]
[35] Elliptic curve cryptography [220]

processes simultaneously use these memory locations to read and write data. As a result, TPMs support standardised encryption schemes (*e.g.* AES[36] in TPMs v1.2 [206] and v2 [210]) and key derivation functions to support secure storage. For each instance, a data value gets encrypted with a storage key (SK), which is derived from the Storage Root Key (SRK) marking the *root of trust for storage* [303].

In storing data pages, it can be distinguished between *binding* and *sealing* of such pages. The process of binding allows the transfer of a storage key, SK_{p_i}, that pertains to party p_i, to any other party, p_j (p_j may decrypt data encrypted under this key). Contrary, in the process of sealing, only the party which sealed the data is able to decrypt the data again.

(e) **Isolated Execution.** To provide privacy and confidentiality of a function execution, it is required that an algorithm can be executed without the interference of other malicious code. TC mechanisms that allow the establishment of isolated areas within a party, where other software running on that same party can not interfere with the code executed within this isolated areas, are termed isolated execution mechanisms. These mechanisms can be achieved via hardware or software enforced techniques. Examples include: (a) the use of virtualisation (*e.g.* Intel's Trusted Execution Technology (TXT)[37]); (b) trusted execution environments (TEEs) (*e.g.* ARM Trustzone[38], Flicker [309], Intel SGX); and (c) containers (*e.g.* Docker[39]). These techniques enable confidential and integrity-preserved processing of code within these isolated areas. Moreover, it prevents other applications on the same privilege level and even code with higher privilege, such as the operating system, a virtual machine manager or any other code, from interfering with the algorithms executed in the isolated areas.

Trusted Platform Modules. A Trusted Platform Module (TPM) is an international standard for a dedicated cryptographic processor that enables cryptographically secure operations. For example, a TPM enables secure storage of cryptographic key material within dedicated secure memory, provides operations to seal and bind data pages and offers software attestation to prove its trustworthy behaviour to external verifiers.

The architectural elements of a TPM are standardised by the ISO/IEC 11889 standard [5]. The TCG published specifications of TPMs in three successive versions: TPM v1.1b [204], TPM v1.2 [206, 207, 208] and TPM v2 [210].

As a brief overview[40], TPMs support the following features (some features are only supported by succeeded TPM versions):

- **Secure storage**: (a) limited persistent memory to store long-term cryptographic keys, and (b) versatile memory to store measurements and short-term keys

[36] Advanced Encryption Standard [333]
[37] http://intel.com/txt
[38] http://arm.com/trustzone
[39] https://www.docker.com/
[40] Our overview presents a selective extraction of important features from the following references [204, 206, 207, 208, 210].

- ◇ **Key generation**: random number generator to create cryptographic keys (*e.g.* for RSA keys)
- ◇ **Cryptographic primitives**: (a) random number generator, (b) public-key cryptographic algorithm, (c) cryptographic hash algorithm, (d) digital signature verification and generation algorithm, (e) attestation algorithm, (f) symmetric-key cryptographic algorithm, (g) key generation and (h) key derivation algorithms
- ◇ **Algorithms**: SHA-1, RSA and AES (for TPMs v1.2 and TPMs v2) and SHA-2 and ECC (for TPMs v2)
- ◇ **Authorisation**: HMAC, PCRs and policies
- ◇ **Key hierarchies**: storage (TPMs v1.2) and additionally platform and endorsement (TPMs v2)

By itself, a TPM does not provide trustworthy behaviour — any malicious application could deploy a TPM. However, applications can leverage the features provided by a TPM to establish trust and prove the applications' trustworthy behaviour. Ultimately, it is the software developer's responsibility to utilise TPMs and other technologies (as introduced previously) to build a chain of trust and provide platform integrity and verification. TREs are examples which leverage TPMs to establish trust and prove trustworthy behaviour to external parties.

2.12.2 Preliminaries on Intel Software Guard eXtenstions

The Intel Software Guard Extension suite [29, 310, 253, 232, 388, 128, 444] comprises novel microprocessor architecture instruction set extensions — available with the Skylake microarchitecture and succeeding microarchitectures — and provide for software developers a software development kit [122, 127] including multiple application programming interfaces (APIs), libraries, documentations [125], sample source code and other tools. Utilising this suite, software developers can secure sensitive operations and data within their applications and establish trust to external parties, which, in turn, may share sensitive information with their applications without the risk of leaking information.

In this context, SGX provides the following guarantees, for a party deploying a supporting Intel CPU:

(a) 3rd party software running on that party can not alter[41] the code and data pages of the program that is executed; therefore, ensuring *integrity*.

(b) The party can process encrypted code and data pages, which are only decrypted inside the immediate CPU components; therefore, ensuring *confidentiality*. The code and data pages remain inaccessible[42] to any other software on the party (including the operating system).

[41] Here, to alter means to alter without detection. The party is able to alter the pages, but such alteration is detected.

[42] Here, inaccessible means in a decrypted format to that party. The party is able to access the ciphertext.

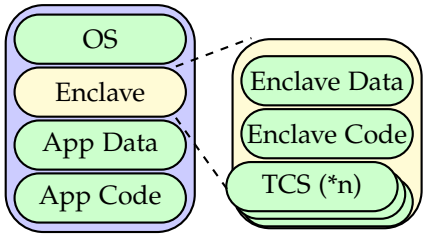


Figure 5: On the left-hand side of a typical user-space process shows the untrusted parts and the trusted enclave. The right-hand side shows the trusted part: an enclave consisting of data pages, code pages and the Thread Control Structure (TCS).

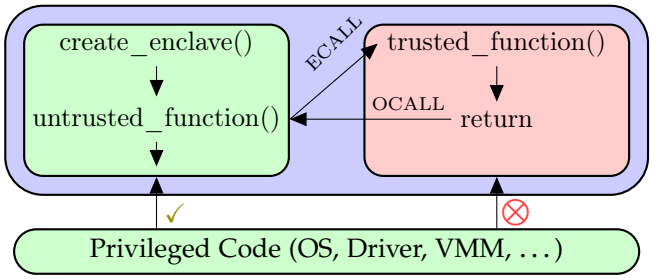


Figure 6: The execution flow of a SGX enclave: at some point after an enclave creation, an ECALL into the enclave's trusted_function() is invoked, which returns after successful execution after the function call (via an OCALL). Any other privileged code has only access to the untrusted user-space application, while access to the trusted enclave is restricted.

SGX is able to provide the above mentioned functionality via so-called secure *enclaves* — small shielded parts of code and data pages within a complex user-space application. (Figure 5 illustrates a user-space process containing an enclave.) Anything within such enclaves is isolated from other software running on the party. SGX enclaves run in the same process as the host application. Access to functions in the same enclave are controlled via ECALL[43] operations, while access to functions outside of the enclave (for example, to the operating system) is controlled via OCALL[44] operations. (Figure 6 illustrates ECALLs and OCALLs of a user-space process and its access via higher-privileged code.) In order to be scalable, SGX allows the creation of multiple secure enclaves within a user-space application. Each enclave has its own isolated memory pages, thus, supporting the principle of least privilege [284, 382] for different parts of the application.

SGX was designed with 8 design principles in mind, which are further outlined in [229, 230, 231]. Since SGX pertains to the category of TC, it also supports the following concepts (similarly as TPMs):

(a) **Isolated Execution.** SGX ensures hardware-enforced isolation of code and data memory pages. Pages, that are marked to be in an enclave can then be accessed only if the call comes from a memory location pertaining to the same enclave. Any access to these pages from other memory locations outside of the enclave are not permitted and result in a signal fault [310]. An enclave is created within

[43] »Enclave call. A function call that enters the enclave.« [127]
[44] »Outside call. A function call that calls an untrusted function from an enclave.« [127]

a user-space process — only this process has access to the memory locations. Higher privilege level software can not circumvent access to these pages since paging is managed by the processor and is never directly accessible to software or other devices. Within the CPU and its immediate components, the pages are decrypted. As soon as a page leaves this environment, for example, the page gets pushed to RAM, it gets encrypted by the Memory Encryption Engine (MEE) [214], using a dedicated custom designed encryption scheme [310].

(b) **Secure Storage.** SGX supports the encrypted storage of large blobs of data via sealing and binding functions. SGX data and code pages are stored in the Enclave Page Cache (EPC) [310]. The EPC natively reserves up to 128 MB of memory space for all enclave pages running on an Intel CPU. To keep an enclave's code base simple, easy to audit for vulnerabilities and to maintain a small TCB (supporting the principle of least privilege), it is recommended to put only sensitive code fragments into an enclave, while storing data pages and non-sensitive code pages on the hard disk. Binding and sealing operations can be used to store pages in an encrypted form (on the hard disk). In a sealing operation, pages are encrypted with a key that is derived from the enclave's identity — MRENCLAVE; pages encrypted with this key can only be decrypted within the same enclave[45]. In a binding operation, pages are encrypted with a key that is derived from the enclave's sealing identity — MRSIGNER; pages encrypted with this key can be decrypted by any enclave pertaining to the same Sealing Authority (SA) [29].

Examples where sealing and binding operations are utilitsed are applications that are provisioned to store long-term data and applications that handle streams of data.

On a related note, with SGX version 1, enclaves are restricted to 128 MB of memory space, since SGX paging is not supported. Since November 2017, with SGX version 2, Enclave Dynamic Memory Management (EDMM) [127, 444] functionality was added, which allows dynamic memory allocation and addition of code and data pages to the enclave at run-time. This feature speeds up enclave creation time by shifting the time to copy pages and measure them to 'on-demand', when additional space is needed.

(c) **Software Attestation.** To establish trust and prove trustworthy behaviour of the functionality included in an enclave, SGX supports local and remote attestation processes [253, 29]. Via local attestation, enclaves can prove themselves to other enclaves running on the same system, while, via remote attestation, an enclave can prove itself to challengers outside of the system (*e.g.* a remote party). In this context, during an enclave's initialisation phase, newly added code and data pages are measured. In the measurement process [29] (via the EEXTEND operation) the contents of an enclave's pages, the relative position of the pages and any security flags associated with the pages are stored in a cryptographic log and a SHA-256 hash over the log is saved to the enclaves identity, MRENCLAVE. For intra-platform attestation, the EREPORT instruction issues a REPORT structure.

[45] MRENCLAVE is a unique enclave identity value. Cryptographic keys derived from this value are bound to this value, thus, it is not possible to decrypt the pages within another enclave.

The verifying enclave takes this report to assess if the proving enclave is a valid structure (indeed, an Intel enclave). For inter-platform attestation, SGX makes use of a special quoting enclave and a group based signatures scheme called Enhanced Privacy ID (EPID) [84]. EPID is a successor to the DAA algorithm [81]. EPID ensures privacy of the issuing CPU, while giving a challenger outside of the party the ability to assess that the enclave is indeed a valid structure.

Additional details about the software attestation process of SGX is given as part of our performance benchmarks in Section 6.6.3. An overview of the security guarantees of SGX is given as part of our threat model in Section 6.5.2. SGX has found widespread application — related works involving SGX include: extensive overview of SGX [128], attestation and sealing [29], software model [310], trustworthy software solutions [232], secure multi-party computation [47], trustworthy data analytics [389], trusted in/outputs [436], memory-safe shielded execution [275], integrity-preserving transparent enclaves [413], secure linux containers [40] and exitless OS services [340] to name a few. A more in-depth overview of research on SGX can be found at Intel's SGX academic research website[46].

2.13 Summary

To extract meaningful information from large data collections, data analysis and data release techniques are applied. These techniques follow a complex multi-step procedure from data collection, pre-processing, data release and data analysis processes and further post-processing to supply various applications with the necessary information in the right format.

In any of these processes, multiple stakeholders are involved that are required to seamlessly cooperate to fulfill the data release/analysis operations to the benefits of all. Yet, sometimes the goals of data providers, data curators and data recipients / analysts differ. Thus, it is important to follow given models. In this context, there are two models: non-interactive (*i.e.* single release) systems and interactive (access is given via a query interface) systems. Following the first model, information is released as a total, while, in the second model, the data resides at the data curator and only partial information is revealed.

To analyse unstructured data and present information in an orderly fashion, various data analysis algorithms may be applied. Importantly though, these algorithms must be optimised to satisfy the requirements of stakeholders such as privacy and data security, scalability, efficiency, utility, and flexibility constraints.

This dissertation is focused on the field of *privacy-preserving* data release and data analysis. This subfield restricts the previous processes: "to enable a party to perform any collaborative operation without diminishing the privacy of an individual, and an individual within a collection" (see Definition 7). Different trust assumptions, models and requirements and constraints of stakeholders lead to a wide range of current practices and techniques from secure multi-party computation to vertically and horizontally-partitioned operations.

[46] https://software.intel.com/en-us/sgx/academic-research

Privacy models set out the privacy constraints; this part introduced syntactic and probabilistic models. The former impose a given structure on the data, while the latter is based on uncertainty at an individual level. In this context, previous authors have introduced modern privacy axioms to ensure internal consistency and standardise privacy models: the axiom of transformation invariance and the axiom of convexity.

Current privacy models can be attributed to three major strands: syntactic privacy models such as k-anonymity and ℓ-diversity; probabilistic privacy models such as differential privacy; and cryptographic privacy models such as secure multi-party computation protocols.

While the privacy model sets out the structure and constraints, privacy algorithms, mechanisms and operations describe how to transform plain data into the required format. Privacy operations are high-level descriptions of a privacy transformation and follow an aggregation or randomisation approach. Privacy mechanism describe the mathematical abstraction and a privacy algorithm comprises a set of instructions to transform the data.

To this end, we discussed techniques to 'measure' the privacy impact of an privacy transformation: information metrics. These metrics can be divided into data and search metrics and follow either a general purpose, a special purpose or a trade-off approach.

Another approach to guarantee input privacy is via secure multi-party computation schemes. The cryptographic approach evolved in the 1980s and 1990s from purely theoretical results to real-world frameworks nowadays. Multi-party protocols operate in two-settings: two-party and multi-party; and take advantage of basic building blocks of modern cryptography such as homomorphic encryption schemes, commitment schemes and zero-knowledge proofs.

Trusted Computing almost seems contrary to multi-party computation, as it focuses on building a trust relationship without the need of fancy cryptographic primitives. In this context, it builds upon the behaviour, and the verifiability of this behaviour, that underly its building blocks which comprise: software attestation, secure storage and isolated execution. Trusted Platform Modules and the SGX are typical implementations of the Trusted Computing domain, whereby SGX benefits from executing directly within the main CPU package.

This part concludes the background information on privacy. The next part of this dissertation introduces privacy notions, their abstraction into privacy games, and their application. Via these games, we aim to clarify understanding of, and relationships between, different privacy notions; we also aim to give an unambiguous understanding of adversarial actions. The games follow the model of privacy-preserving data release and data analysis discussed in this part; as an example, we apply the games to a specialisation of data analysis tasks: recommender systems.

Part I

Representation and Modelling of Privacy

» *Our work to improve privacy continues today.* «

— Mark Zuckerberg, CEO of Facebook

3 ABSTRACTING SYNTACTIC PRIVACY NOTIONS VIA GAMES

Contents

Publication Data

This chapter is based on the following publication:

(1) Robin Ankele and Andrew Simpson. Abstracting Syntactic Privacy Notions via Privacy Games. In: Proceedings of ATC 2019. 2019. [35]

In this chapter, we present privacy games as a simplified abstraction of syntactic privacy notions. We start by introducing various syntactic privacy notions from previous authors. Then, we present a system and privacy model that extends previous contributions. Next, we present a short analysis of previous contributions and our extension of a semi-distributed privacy model. In this context, we reveal new insights and relationships between privacy notions. Then, we present our game-based abstractions of the privacy notions. We start by outlining preliminaries of our abstraction to privacy games: (a) our adversarial model and (b) principles of modelling privacy games. Our main contribution of this chapter is the modelling of privacy games for individual and group privacy within distributed environments. For each selected privacy notion, we formulate a game that an adversary has to win in order to break the notion. Via these games, we aim to clarify understanding of, and relationships between, different privacy notions; we also aim to give an unambiguous understanding of adversarial actions. The abstraction as privacy games should enable experts to obtain an alternative viewpoint (to the purely textual representation of other contributions). Moreover, the agglomeration of the contributions of previous authors should help to combine these endeavors and foster the generation of new insights and enable

better analyses of privacy-preserving applications. Overall, we aim to foster a better understanding of privacy, its meaning, implications and relevant notions. Finally, we summarise the contributions of this chapter.

3.1 Overview

To provide a clear understanding of privacy, its properties and implications for systems, it is important to accurately model adversarial actions. In recent years, several such approaches have been defined; examples include:

(a) **Behavioural**: Adversaries may operate adaptively or in a fixed manner[1] [290]. Further, an adversary may behave in a semi-honest, malicious or covert fashion [290, 223, 43]. As such, a semi-honest adversary strictly follows the defined protocol, a malicious adversary deviates arbitrarily at any point, and a covert adversary deviates only when deemed to remain undetected.

(b) **Game-theoretic**: Modelling adversaries in a game-theoretic fashion allows reasoning about the strategy of each player in a protocol and captures real-life behaviour such as collaboration between certain parties. Examples include [259, 320, 112].

(c) **Simulation-based**: In this setting, notions are modelled in an ideal world (the simulator) and the real world (the protocol). An adversary is given access to both oracles and wins if she is able to distinguish between them with a non-negligible advantage. This approach was introduced by Goldreich *et al.* [195]; further, Lindell [288] provides an excellent tutorial for this technique.

(d) **Game-based**: Game-based security notions have been widely used in the cryptography community to provide simpler and more easily verifiable proofs. For this technique [55] notions are formulated in games illustrating the necessary actions for an adversary to break said notion. Examples include [55, 15]. Moreover, such games are also widely used to reason about privacy properties (for example, within anonymous authentication protocols [287, 28]). In this context, games abstract notions of unforgeability, seclusiveness, anonymity or unlinkability in pseudonymous signatures schemes [270] or attribute-based credential schemes [373].

In this chapter, we focus on the last technique — as such, we present games for a variety of privacy notions. In this context, within recent years a variety of privacy notions have been proposed, some of them being of syntactic nature, others of probabilistic nature. These variety of notions capture different perspectives on and properties of privacy[2] — leaving non-experts in an unclear situation with regards to selecting a 'fitting' privacy notion.

[1] By 'fixed manner' we understand that an adversary, prior to attacking a system, has fixed its strategy on how she is going to attack the system and does not change her strategy with regards to results she obtains from the system under attack.

[2] Differential privacy [159, 160], for example, is a property of the release algorithm and, as such, independent of the processed data. The level of privacy guarantees / data utility that can be achieved depends on the

Frameworks that aim to unify (*e.g.* clarify understanding of) and simplify (*e.g.* show relationships between) different privacy notions have been proposed by various authors. For example, (mainly focused on probabilistic privacy) Kifer and Machanavajjhala present the Pufferfish framework [265] and Li *et al.* present the Membership privacy framework [283]. (Definitions within these frameworks satisfy the fundamental axioms for modern privacy design guidelines: *transformation invariance* and *convexity* [263].) Domingo-Ferrer [153] proposed the notion of co-privacy, a game-theoretic approach, with the attractive property that the best strategy of a player is to help preserve the privacy of other players. A similar approach was defined by Sánchez *et al.* [385], who proposed the notion of co-utility. In co-utile protocols, a player increases her utility by helping others to increase theirs. Privacy games defined in these frameworks aim to identify the 'best' strategy for their players to receive the best outcome. A player's incentive is to follow this strategy, thus enhancing privacy/utility of the overall protocol outcome. Alternatively, the games defined in our context are aimed to provide experts with a better understanding of the underlying privacy notions.

In the case of syntactic privacy notions, Pfitzmann and Köhntopp established a consistent terminology [358] with the aim to unify said notions. Despite such efforts, an open problem with the formal treatment of syntactic privacy notions remained: discrepancies between formal models led to ill-defined relationships and incomparable notions. In addition, various attacks [298, 282, 188, 439, 262] influenced and challenged the formulation of new privacy notions. Bohli and Pashalidis addressed those disconnected notions in [72], as a first step to unite multiple privacy notions into a single formal framework.

In this chapter, we build upon the privacy notions of [358] and [72]: as such, for each selected privacy notion, we formulate a game an adversary has to win in order to break the notion. Via these games, we aim to clarify understanding of, and relationships between, different privacy notions; we also aim to give an *unambiguous understanding* of adversarial actions; moreover, via the games we aim to foster analyses of privacy notions and the generation of new insights.

Nevertheless, we note, while the presented notions and games are capable of addressing a wide range of privacy abstractions, not all issues may be covered by our definitions. As such, some restrictions are inherent from our system model and our adversary model. Other limitations and relationships between notions and our proposed games are further discussed in the next chapter.

3.2 Privacy Notions

In the following, we provide an overview of the privacy notions of Pfitzmann and Köhntopp [358] and Bohli and Pashalidis [72]. As a disclaimer, the notions of [358] represent privacy from a viewpoint of an entity within a system (*e.g.* a subject, an

sensitivity of the query function. Differential privacy is an inherently probabilistic approach. Alternatively, syntactic privacy notions (including our privacy games) are dependent on the data. Concretely, privacy notions are defined on the relations between data elements. The level of privacy guarantees depends on how a release algorithm can obfuscate these relationships.

object, an action), while the notions of [72] represent privacy from the viewpoint of a system (*i.e.* a system provides notion x if condition y holds).

3.2.1 Privacy Notions by Pfitzmann and Köhntopp

The exact meaning of 'providing privacy' is strongly dependent on the setting and the type of system to be evaluated. Unfortunately, due to the varying nature of privacy definitions, there have been several discrepancies in the definition of a consistent terminology. Consequently, in [358] Pfitzmann and Köhntopp propose a general terminology uniting several privacy definitions.

Pfitzmann and Köhntopp define sender and recipient of messages in a system as *subjects*. Messages sent throughout the communication system are referred as *objects*. Operations such as sending or receiving a message are defined as *actions*.

Then, [358] defines the following privacy notions (we provide only a brief overview of these privacy notions; for a detailed description of the notions we refer to [358]):

Definition 18 (Anonymity)**.** *A subject is anonymous if an adversary cannot sufficiently identify the subject within a set of subjects (the anonymity set).*

Definition 19 (Unlinkability)**.** *Two ore more items of interest (e.g. subject, object or action) are unlinkable if an adversary cannot sufficiently distinguish whether those items are related or not.*

Definition 20 (Undetectability)**.** *An item of interest (e.g. subject, object or action) is undetectable if an adversary cannot sufficiently distinguish if the item exists or not.*

Definition 21 (Unobservability)**.** *An item of interest (e.g. subject, object or action) is unobservable if it provides undetectability against all subjects uninvolved in it and anonymity of the subject involved in it, even against the other subjects involved in it.*

Definition 22 (Pseudonymity)**.** *Pseudonymity is the use of pseudonyms as identifiers. Consequently, a pseudonym is an identifier of the subject.*

In their definitions, Pfitzmann and Köhntopp recognise that the wording 'sufficiently' indicates that the properties (*i.e.* characterisations of privacy) are quantifiable and (for certain applications) there is a threshold when the property holds (or doesn't).

3.2.2 Privacy Notions by Bohli and Pashalidis

In order to address and unify privacy notions in a single formal framework (to compare and reason about privacy properties in a formal way) Bohli and Pashalidis present in [72] a framework with unambiguous privacy notions. We first introduce each of their proposed notions in terms of natural language (to aid comprehension), and then state a more formal definition.

Before introducing our system model in detail in Section 3.4, we briefly state some definitions to follow the privacy notions below. We define a mapping function f, which maps a user identifier u_i to an element e_i released from the system. Moreover, over the set of all released elements, we define the following sets: U_f, the participation set,

contains all user identifiers; Q_f, the user frequency set, contains a mapping of user identifiers and number of elements that map to that user identifier; and P_f, the linking relation, partitions the released elements with regards to their user identifiers.

Informally, we say that an adversary (simplified) is allowed to interact[3] with a system as follows:

(a) it inputs information, and/or

(b) it obtains an outcome — in our case, a vector of input values (transformed (*e.g.* permutated or perturbed) to ensure privacy) and an aggregated value (computed over the input values).

Each privacy notion embodies its own restrictions or requirements on the released data from a system. Table 4 summarises the notions we consider in this chapter. The notions are listed with respect to a partial order dependent on the knowledge of an adversary, $\mathcal{A}$ (and its access to certain interfaces). In natural words, the objectives of the privacy notions introduced by Bohli and Pashalidis [72] are as follows (in addition, Figure 15 further illustrates the privacy notions in terms of our system model):

(a) For **anonymity**, an adversary should not be able to learn any interesting relationship between individual or groups of outcome elements (especially the user identifier) with regards to the input elements.

Bohli and Pashalidis formally define the notions of strong, standard, and weak anonymity (Def. 23-25) as follows:

Definition 23 (Strong Anonymity)**.** *A system is said to provide strong anonymity, denoted by* SA*, if it does not leak any information about the mapping function* f*.*

Definition 24 (Anonymity)**.** *A system is said to provide anonymity, denoted by* AN*, if it does not leak any information about* f *beyond the participation set* U_f *and the linking relation* P_f*.*

Definition 25 (Weak Anonymity)**.** *A system is said to provide weak anonymity, denoted by* WA*, if it does not leak any information about* f *beyond the participation set* U_f*, the usage frequency set* Q_f *and the linking relation* P_f*.*

For the notion of strong anonymity (Def. 23) an adversary must not learn any information on the relation between a subject and objects. For standard anonymity (Def. 24), an adversary is allowed to learn the subjects and objects, but not how these relate to each other. For weak anonymity (Def. 25), an adversary is allowed to learn the subjects, objects and the number of objects corresponding to each subject, but not how these relate to each other.

(b) For **participation hiding**, an adversary should not be able to link an outcome element to a specific user identifier.

Formally, participation hiding is defined as:

Definition 26 (Participation Hiding)**.** *A system is said to hide participation, denoted by* PH*, if it does not leak any information about* f *beyond the size of the participant set* $|U_f|$*.*

[3] This is not a one-shot interaction, but a user (or adversary) may access the system interfaces multiple times.

In a system that provides participation hiding, an adversary is not allowed to gain any more information than the number of subjects participating in the evaluation of the system.

(c) In the case of **unlinkability**, an adversary should not be able to link released elements pertaining to the same user identifier.

Bohli and Pashalidis formally define the notions of strong and weak unlinkability as follows:

Definition 27 (Strong Unlinkability)**.** *A system is said to provide strong unlinkability, denoted by* SU*, if it does not leak any information about* f *beyond the participant set* U_f*.*

Definition 28 (Weak Unlinkability)**.** *A system is said to provide weak unlinkability, denoted by* WU*, if it does not leak any information about* f *beyond the usage frequency set* Q_f[4]*.*

A system achieving the notion of strong unlinkability (Def. 27) does not leak any more information to the adversary beyond the list of subjects participating in the system. Furthermore, a system achieving the notion of weak unlinkability (Def. 28) does not leak any more information than the number of objects corresponding to each subject.

(d) For **pseudonymity**, we may allow an adversary to link element groups with each other; however, we aim to prevent her from learning the actual user identifier, though allowing her to assign that group with a pseudonym.

Formally, pseudonymity is defined as:

Definition 29 (Pseudonymity)**.** *A system is said to provide pseudonymity, denoted by* PS*, if it does not leak any information about* f *beyond the linking relation* P_f*.*

Bohli and Pashalidis define pseudonymity (Def. 29) as a notion where the adversary is able to gain no more information than the linking relation between objects. In this setting, each object can be linked to a pseudonym, but not the real name of the subject.

3.3 Systems

Many privacy models (such as [408], [298], [282] and [159][5]) tend to assume that data to be shared or analysed originates from a centralised database. In this context, users must always trust the data curator to keep their data secure (and, consequently, preserve their privacy). Such models, however, do not capture use cases associated with distributed environments — in which data remains split over several devices and is stored in databases *distributed* over a large network. Yet, constant improvements in technology allow increasingly more operations to be executed in distributed environments (which seeks to preserve the privacy of those individuals involved in such processing). In the following, we start by giving a generic introduction to systems (used

[4]Note that knowledge of Q_f implicitly implies knowledge of U_f.

[5]We note that there exist variants of said privacy models which operate in a distributed setting. However, typically, these models operate on a centralised database.

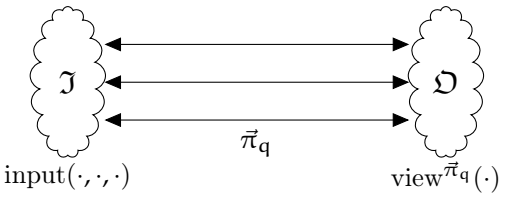


Figure 7: Mathematical abstraction of a system: a set of transformation functions $\vec{\pi}_q$ maps values from the input space $\mathfrak{I}$ to an output space $\mathfrak{O}$. Users interact with the system via interfaces (for example, input$(\cdot,\cdot,\cdot)$ and view$^{\vec{\pi}_q}(\cdot)$).

to abstract an application) and, then, illustrate the differences between centralised and distributed systems.

3.3.1 Generic Definitions of Systems

We describe 'black box' models of systems which we use as a generic abstraction to argue about privacy properties for various applications. We aim to simplify such an abstraction to the bare minimum (in order to remain as generic as possible). As such, we define a system as follows:

Definition 30 (System)**.** *A system, denoted as ϕ, takes a finite number of parameters as input, which are passed through a finite set of transition functions which, thereafter, release a finite number of parameters.*

A system takes a finite number of input parameters α_i (in vector notation $\vec{\alpha}$ to denote the whole input to a system) and releases output parameters e_i ($\vec{e}$). The input and output parameters are drawn from input and output spaces, denoted $\mathfrak{I}$ and $\mathfrak{O}$ respectively. The mapping from the input space to the output space is performed via a finite set of transformation functions $\vec{\pi}_q$ ($\vec{\pi}_{q,i}$s may be composite in the form of $\pi_{q,1} \circ \pi_{q,2} \circ \cdots \circ \pi_{q,n}$). This abstraction (of a system) is further illustrated in Figure 7.

With such a definition of a system, we are able to abstract various real-world applications. We are in particular interested in systems that give us some sort of privacy guarantees. Therefore, we add two more definitions for systems: the first with regards to *input privacy* and the second with regards to *release privacy*[6].

Definition 31 (Input Privacy-Preserving Systems)**.** *A system ϕ is said to be computationally-bounded privacy-preserving, with respect to input privacy, if it guarantees with any sort of mechanism that the execution and internal communication of input parameters (throughout a protocol) does not leak any information to an adversary, $\mathcal{A}$.*

Definition 32 (Release Privacy-Preserving Systems)**.** *A system ϕ is said to be computationally-bounded privacy-preserving*[7]*, with respect to release privacy, if it guarantees*

[6]There are various definitions for privacy, and similar properties that arguably can be included in the context of privacy. We focus on those definitions outlined in Section 3.2.

[7]We distinguish between computationally-bounded privacy-preservation and perfect privacy-preservation. In the former case, privacy may be preserved up to a probabilistic bound (preferable, infinitesimal small). In the latter case, even a computationally-unbounded adversary must not be able to infer knowledge in order to break the privacy guarantees given by the system.

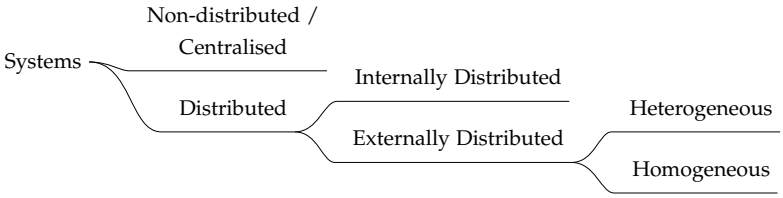


Figure 8: Simplified taxonomy of systems, which we consider in this dissertation.

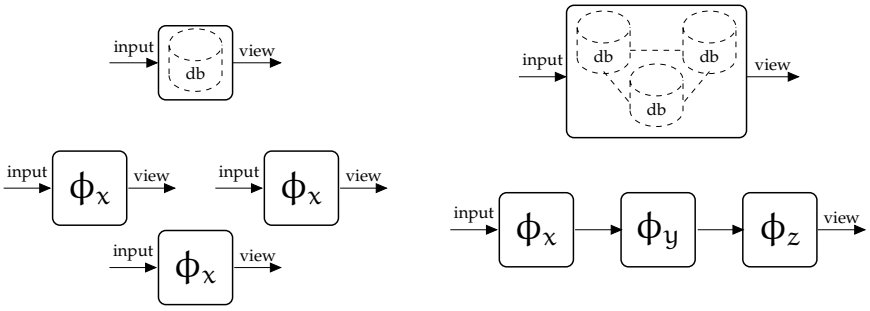


Figure 9: Illustration of the architecture of systems: a centralised system (top-left), an internally distributed system (top right), a homogeneous system (bottom left) and a heterogeneous system (bottom right).

with any sort of mechanism that an adversary $\mathcal{A}$ is not able to infer the input parameters (or, certain properties about the input parameters) from the observation of the output parameters.

3.3.2 Classification of Systems

Systems may be classified according to various properties. These include system behaviour, security or privacy guarantees and the applicability to certain use cases. We establish a simplified taxonomy of systems given in Figure 8 according to potential architectural features. An illustrative overview of these system classes is further given in Figure 9.

(a) **Non-distributed or Centralised systems:** Many systems are abstracted in a centralised way due to its simplistic interfaces: users typically gain access to a single input and view interface in order to interact with the system. In general, the characteristics of such a model further assume that data is to be held in a single dataset. Thus, any internal handling is abstracted from the outside; in other words, we can see such a model as a black box. Any internal handling is assumed to be secure and privacy-preserving, while a system designer is concerned with preserving the privacy of the released data, and any information inferable from this as a whole. An example of such systems is privacy-preserving deep learning [10].

(b) **Distributed systems:** Systems that underlie a distributed model are assumed to store their data in distributed databases (split over several subsystems). Users interact with these systems still via input and view interfaces, yet, each system provides its own interface. Examples of systems underlying a distributed archi-

tecture include crowdsourcing statistics [176], federated learning[8] [311], or usage patterns collections (for example, the approach by Apple[9] in iOS 10).

In this context, in our abstraction, we consider the following types of distributed systems:

(a) **Internally distributed systems:** Systems of this kind still consider the system release as a whole: as such, privacy properties are defined on the whole release. Such systems differ from centralised systems in the following way: internally distributed systems consider data to be distributed among various parties (*i.e.* each of these parties holds a separate database), while in centralised systems only one party holds the whole database. Although, similarly as in centralised systems, users (and, with that, also potential adversaries) interact via a single input and view interface.

(b) **Externally distributed systems:** Systems of this kind split the system release among a collection of systems. In other words, externally distributed systems represent a collection of single systems. The privacy properties are manifested in each system's individual release and in the combined release. Therefore (with the definition and evaluation of the privacy notions), it must be taken into consideration that an adversary is able to learn information from each single system, but also information from their combination.

External distributed systems may be further specialised into the classes of homogeneous and heterogeneous systems as described below:

(a) **Homogeneous systems:** In such a collection of systems, each individual system executes the same transformation function, π_q. Users get access to each system's input and view interface to interact with the collection of systems. The released information of each system may be combined to generate a system-wide release.

(b) **Heterogeneous systems:** In such a collection of systems, each individual system is to execute a different transformation, $\pi_{q,i}$. Moreover, we assume that each system's release is the next system's input, therefore creating a chain of systems. Users may get access to each system's input and view interface or only to the first system's input and the last system's view interface.

3.4 A Formal Characterisation of our System Model

In this section, we present a formal characterisation of our system model. Our characterisation extends (with respect to distributed systems) the system model of Bohli and Pashalidis [72].

We consider systems, denoted by ϕ, that are sequentially invoked a finite number of times, and require that they return values $e_i \in \vec{e}$ (where $\vec{e}$ denotes the set of all return values for a particular release) and an aggregated value β computed over a

[8] https://research.googleblog.com/2017/04/federated-learning-collaborative.html
[9] https://wired.com/2016/06/apples-differential-privacy-collecting-data

finite number of inputs $\alpha_i \in \vec{\alpha}$ (similarly, $\vec{\alpha}$ denotes the set of all input values for a particular release). Further, we require that each input α_i must be associated to a unique user identifier u_i and a computational party p_i. Inputs α_i associated to the same user identifier u_i may also collectively be denoted in vector notation $\vec{\alpha}_{u_i}$, with $\alpha_{u_i,i}$ denoting the value α_i within $\vec{\alpha}_{u_i}$. The same applies for the output values e_i, with $\vec{e}_{u_i}$ for outputs associated to the same u_i and $e_{u_i,i}$ denoting the value e_i within $\vec{e}_{u_i}$. Tuples of the form (u_i, α_i) are stored within datasets at the specified party p_i. The inputs to, as well as the outputs from, ϕ are drawn from parameter spaces $\alpha_i \in \mathfrak{I}, e_i \in \mathfrak{O}$, where $\mathfrak{I}$ denotes the input space, $\mathfrak{O}$ denotes the output space and any values of $\mathfrak{I}$ and $\mathfrak{O}$ are specific to the system ϕ. (Dependent on the system, the input and output space may be equivalent, *i.e.* $\mathfrak{I} \equiv \mathfrak{O}$.) Similarly defined is the parameter space of the aggregated output $\beta \in \mathfrak{O}_\beta$. Each user identifier u_i and party identifier p_i is assumed to be unique, and drawn from an identifier space.

Users interact with systems via interfaces — for example, an interface, $\mathrm{input}(\cdot,\cdot,\cdot)$, to process inputs to the system and an interface, $\mathrm{view}^{\vec{\pi}_q}(\cdot)$, to receive the system release. Invocations of ϕ, in the form of a query q_i, produce output batches $(\vec{e}_i, \beta_i)$[10] of potentially varying size. In detail, a set of transformation functions, denoted by $\vec{\pi}_q$, map a value, α_i (or a set of values, $\alpha_i \in \vec{\alpha}$), to an outcome value, e_i, as well as a single aggregated value, β. The length of the outcome parameters is not necessarily the same as the length of the input parameters. (Formally, $|\vec{\alpha}| = |\vec{e}|$ or $|\vec{\alpha}| \neq |\vec{e}|$; in particular, such a mapping may be non-injective[11].) A transformation function, $\pi_{q,i}$, is defined as follows:

$$\pi_{q,i} : \mathfrak{I} \mapsto (\mathfrak{O}, \mathfrak{O}_\beta); \vec{\alpha}_i \mapsto \pi_{q,i}(\vec{\alpha}_i) \tag{1}$$

with $(\vec{e}_i, \beta_i) = \pi_{q,i}(\vec{\alpha}_i)$, $\vec{\alpha}_i \subseteq \vec{\alpha}$, $\vec{e}_i \subseteq \vec{e}$, $\beta_i \vdash$ [12]β, $\mathfrak{I}$ the input parameter space and $\mathfrak{O}$ and $\mathfrak{O}_\beta$ the output parameter spaces. A set of transformation function, $\vec{\pi}_q$, satisfy the following properties:

(i) composition:

$$\vec{\pi}_q = \pi_{q,1} \circ \pi_{q,2} \circ \cdots \circ \pi_{q,n} \tag{2}$$

where $(\vec{e}, \beta) = \vec{\pi}_q(\vec{\alpha})$ and for any subsets $\vec{\alpha}_i \in \vec{\alpha}$ a transformation function $(\vec{e}_i, \beta_i) = \pi_{q,i}(\vec{\alpha}_i)$ applies.

(ii) chaining:

$$(\vec{e}, \beta) = \vec{\pi}_{q,n}(\vec{\pi}_{q,n-1}(\cdots \ \vec{\pi}_{q,1}(\alpha))) \tag{3}$$

where any $\vec{\pi}_{q,i}$ takes as input the released values of $\vec{\pi}_{q,i-1}$ and inputs to $\vec{\pi}_{q,1}$ are $\alpha_i \in \vec{\alpha}$ and release elements of $\vec{\pi}_{q,n}$ are $(\vec{e}, \beta)$.

[10] When we write $(\vec{e}_i, \beta_i)$, we indicate a specific query q_i. Otherwise we use $(\vec{e}, \beta)$.

[11] We do not require the mapping to be surjective, since this would mean that every 'possible' element of the codomain is mapped to by at least one element of the domain. Yet, we require that every 'actual' output element must be mappable (through inversion of the privacy-preserving function) to an input element.

[12] We use the notation $\vdash$ to indicate that β_i is a part of the element β. Concretely, the single value β is composite: $\beta_1 \circ \beta_2 \circ \cdots \circ \beta_n$, where $\circ$ is an operation that combines these values into one single value. For example, $\circ$ may be the $+$ operation.

Property (i) (composition) is especially useful in the context of single, internally distributed systems (*e.g.* semi-distributed systems) where each party $p_i \in P$ executes a transformation function $\pi_{q,i}$ on a subset of the input data $\vec{\alpha}_i \subseteq \vec{\alpha}$. Property (ii) (chaining) is useful in the context of multiple (distributed) systems, where each party p_i executes a set of (potentially different) transformation functions $\vec{\pi}_{q,i}$, which can be executed in a chain.

A query q_i (to system ϕ) comprises a set of transformation functions $\vec{\pi}_q$: these functions operate on the input data of a system, ϕ; the query result is the outcome of what is released by the last transformation function (*i.e.* $(\vec{e}_{\vec{\pi}_{q,n}}, \beta_{\vec{\pi}_{q,n}})$ of $\vec{\pi}_{q,n}$; simplified we denote the outcome as $(\vec{e}, \beta)$).

Overall, ϕ accepts input batches of the form

$$(u_1, \alpha_1, p_1), (u_2, \alpha_2, p_2), \ldots, (u_c, \alpha_c, p_c) \tag{4}$$

for c invocations of the input interface $\text{input}(\cdot, \cdot, \cdot)$. In order to preserve a user u_i's privacy, for each invocation of the output interface $\text{view}^{\vec{\pi}_q}(\cdot)$, ϕ applies a set of privacy-preserving transformation functions $\vec{\pi}_q$. $\pi_{q,i}$s may be, for example, secret permutations. In detail, for each party p_i, a $\pi_{q,i}$ is applied to its datasets. Taken together (aggregating all results of the $\pi_{q,i}$s), ϕ outputs a sequence

$$(e_1, \ldots, e_c), \beta. \tag{5}$$

Note that we do not limit $\pi_{q,i}$ to be a secret permutation, but $\pi_{q,i}$ may be any privacy-preserving mapping function. An illustration of the mapping of a transformation function is given in Figure 10.

A system, ϕ, may consist of a single party, p, or a set of parties, $p_i \in \vec{p}$, in order to store and process data. Essentially, a system is an abstraction of a party or set of parties; the parties are the engines to store, represent and operate on the data. In the notation above, we assumed that each input (u_i, α_i) is distributed to a different party p_i. However, we do not limit our model in such contexts, but allow a data owner[13] to freely decide to which party p_i she wants to send her inputs. Our model is assumed to handle any data distribution among parties p_i.

Building on the privacy model of Bohli and Pashalidis [72], our model includes the possibility of a 'void' invocation of the system, *i.e.* an invocation of ϕ, where the outcome $(\vec{e}, \beta)$ is unrelated to any u_i, α_i or p_i and drawn uniformly at random from the parameter space $\mathfrak{O}$. However, we require that the outcome of a void invocation must remain indistinguishable to the outcome of a 'normal' invocation of ϕ. This property is essential in the context of system unobservability. Our model allows single users or a collection of users to reason about their privacy assurances. Precisely, a notion or game can be applied to capture privacy properties about a mapping between one of:

1. a single user and an attribute: $u_x \rightarrow e_{u_x}$

[13] Data submitted to the system may include (or pertain to) several data subjects, that are independent of the actual data owner. In our modelling, so far, we assume that each input value α_i is associated to a user identifier u_i, which identifies the data subject (which may or may not be the data owner). We further assume that data subjects have given consent to data owners to perform certain data-mining tasks on the committed data, while preserving the privacy of the data subjects. In any case, the data owner decides with which party p_i to share the data.

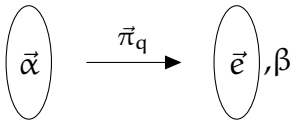


Figure 10: Abstraction of the transformation function $\vec{\pi}_q : \alpha_i \to e_i, \beta_i$: elements $\alpha_i \in \vec{\alpha}$ map to output elements $e_i \in \vec{e}$. $\mathcal{A}$ may know $f^\star(\alpha_i) = u_i$, but must not learn $f(e_i) = u_i$.

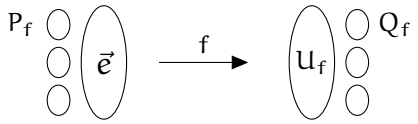


Figure 11: Abstraction of the mapping function $f : e_i \to u_i$: each element $e_i \in \vec{e}$ maps to a unique user identifier $u_i \in U_f$. P_f partitions $\vec{e}$ into disjoint subsets; likewise, Q_f partitions U_f into disjoint subsets.

2. a single user and multiple attributes: $u_x \to (e_{u_x,1}, \ldots, e_{u_x,c})$
3. multiple users and multiple attributes: $u_x \to e_{u_x}, u_y \to (e_{u_y,1}, \ldots, e_{u_y,c})$

where the last case denotes group privacy.

In order to abstract privacy properties for systems, Bohli and Pashalidis [72] have, for their privacy notions, defined a function f, which associates the user identifier, u_i, with the released elements, e_i. This association is seen to be *sensitive* if learned by an adversary, $\mathcal{A}$. Using privacy mechanisms as part of the set of transformation function, $\vec{\pi}_q$, may prevent such information gain. In addition, the function $f^\star$ maps an input element, α_i, to the pertaining user identifier, u_i. Depending on the adversarial model, an adversary, $\mathcal{A}$, may learn this relationship.

More formally, we define the functions f and $f^\star$ as follows. First, f

$$f : \mathfrak{O} \mapsto \mathfrak{U}; e_i \mapsto f(e_i) \tag{6}$$

with $u_i = f(e_i)$, $\mathfrak{O}$ the parameter space of the outcome elements, and $\mathfrak{U}$ the parameter space of the user identifiers, takes as an input an element, e_i, (or vector $\vec{e}$[14]) of ϕ and maps it to the corresponding user u_i. Similarly, $f^\star$

$$f^\star : \mathfrak{I} \mapsto \mathfrak{U}; \alpha_i \mapsto f(\alpha_i) \tag{7}$$

with $u_i = f(\alpha_i)$, $\mathfrak{I}$ the parameter space of the input elements, and $\mathfrak{U}$ the parameter space of the user identifiers, takes as an input an element, α_i, (or vector $\vec{\alpha}$) (stored in the dataset of any party p_i) of ϕ and maps it to the corresponding user u_i.

Through the processing of a query q_i (*i.e.* essentially through any $\pi_{q,i}$), ϕ obfuscates the function f. An adversary $\mathcal{A}$ must not be able to directly learn f by observing the inputs (u_i, α_i, p_i) to ϕ or the released outcome $(\vec{e}, \beta)$. Nevertheless, the adversary's goal remains to determine f, or any other interesting property about f. Still, the privacy notions and games reveal, to varying degrees, information about f.

Figure 11 further illustrates the abstraction of the mapping function f. Moreover, it indicates how the sets U_f, Q_f and P_f are associated to f.

We denote $p_i \in P$, with P the set of all computational parties. Inputs of ϕ are stored in datasets $\alpha_i \to p_i.db$, given any possible distribution. Moreover, we say $\vec{\alpha} = \cup_{i=1}^{P} p_i.db$, that is, $\vec{\alpha}$ is the set of all inputs $\alpha_i \in \vec{\alpha}$ for all parties p_i. Similarly, it holds that $\vec{e} = \cup_{i=1}^{n} e_i$ represents the union of all outcome values.

[14] For simplicity, we allow the notation of $f(\vec{e})$ and mean that the function f is called i times; $\vec{u} = f(\vec{e})$ releases a vector of user identifiers with $u_i \in \vec{u}$ (in addition, $f(\vec{e}) \equiv \{\forall x, 0 \leqslant x \leqslant i : f(e_x)\}$).

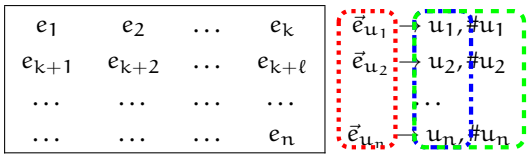


Figure 12: The viewpoint of an adversary, $\mathcal{A}$: the sets $\mathbf{U_f}$ (- - -), $\mathbf{Q_f}$ (- - -) and $\mathbf{P_f}$ (- - -) are defined on the released outcome elements $\vec{e}$. By gaining access to the sets, $\mathcal{A}$ gains knowledge to identify the mapping function f.

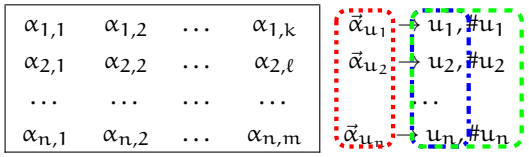


Figure 13: The systems internal viewpoint: the sets $\mathbf{U_f^\star}$ (- - -), $\mathbf{Q_f^\star}$ (- - -) and $\mathbf{P_f^\star}$ (- - -) are defined on the input elements $\vec{\alpha}$.

Following from the definitions of Bohli and Pashalidis [72], we denote the following properties of the outcome in the form of sets:

Definition 33 (Participant Set). $U_f = \{f(e_i) : e_i \in \vec{e}\}$ *denotes the set of participants in* ϕ. *Each element* $e_i \in \vec{e}$ *is uniquely associated to a user identifier* $f(e_i) = u_i$.

Definition 34 (Usage Frequency Set). $Q_f = \{(u_i, \#u_i) : u_i \in U_f\}$, *with* $\#u_i = |e_i \in \vec{e} : f(e_i) = u_i|$, *denotes the relation of the number of elements* $e_i \in \vec{e}$ *each user* u_i *is associated with. In other words,* $\#u_i$ *is the number of* e_i *user* u_i *has contributed to* ϕ.

Definition 35 (Linking Relation). $P_f = \{\vec{e}_{u_1}, \vec{e}_{u_2}, .., \vec{e}_{u_{|U_f|}}\} \vdash \vec{e}$ *denotes the linking relation of elements* $e_i \in \vec{e}$, *which pertain to a certain user,* u_i. *In other words,* $\vec{e}$ *is partitioned into disjoint subsets,* $\vec{e}_{u_i}$, *where* $\vec{e}_{u_i}$ *consists of all* e_i *for user* u_i. *Precisely, it holds that* $\forall e_i \in \vec{e}_{u_i} : f(e_i) = u_i$.

In Figure 15, we illustrate the relationship between our system model and the previously defined privacy notions (see Table 4 for an overview). As depicted, the system ϕ is queried with a triple (u_x, α_x, p_i) and releases a pair $(\vec{e}, \beta)$ on a view query. Dashed encircled are the sets that the adversary $\mathcal{A}$ is given access to for each notion. Further, the dashed arrows show the objective of each notion. For simplicity, this representation includes only one database: $p_i.db$. However, any finite number of databases are equally possible and supported by the model.

Figure 12 further illustrates the relationship between the outcome $\vec{e}$ of ϕ and the previously defined sets U_f, Q_f and P_f from the viewpoint of an adversary, $\mathcal{A}$. Moreover, Figure 13 illustrates the relationship of the sets $U_f^\star, Q_f^\star$ and $P_f^\star$ from the internal viewpoint of system ϕ. In general, it holds that $U_f^\star = U_f, Q_f^\star = Q_f$ (dependent on the privacy-preserving function, $\vec{\pi}_q$, the ordering of the sets may be different (*e.g.* $\vec{\pi}_q$ is a permutation)). P_f consists of partitions of $\vec{e}$, while $P_f^\star$ consists of partitions of $\vec{\alpha}$. Dependent on the adversarial model, an adversary, $\mathcal{A}$, might learn (or have access to interfaces) to all, or to a subset, of these sets. Yet, the overall goal of $\mathcal{A}$ is to find function f, which allows her to map user identifiers u_is to outcome elements e_is.

Remark 1. *For a fixed user* u_x*, we denote the subsets* $U_{f_x} \subseteq U_f, Q_{f_x} \subseteq Q_f$ *and* $P_{f_x} \subseteq P_f$*, where* $U_{f_x} = u_x, Q_{f_x} = (u_x, \#u_x)$ *and* $P_{f_x} = \vec{e}_{u_x} = (e_{x,1}, .., e_{x,k})$*. We assume that* $\mathcal{A}$*, by knowledge of* U_{f_x}, Q_{f_x} *and* P_{f_x}*, in combination, can* uniquely *determine (or evaluate) the mapping function* $f(e_x) = u_x$ *and therefore* uniquely *map* $e_x \rightarrow u_x$*. However, this holds only in this limited setting, and with the assumption that* $\mathcal{A}$ *obtains the above sets. By knowledge of the generalised (system-wide) parameters* U_f, Q_f *and* P_f*,* $\mathcal{A}$ *may not be able to uniquely map* $f(e_x) = u_x$*, due to the fact that equivalence classes in* Q_f *do not have to be unique. For example, if the input distribution is uniform (i.e. each user contributes the same number of input values to the system),* $\mathcal{A}$ *would not gain any knowledge about the mapping of* $f(e_x) = u_x$ *by solely observing* U_f, Q_f *and* P_f*.*

3.5 Analysis of the System and Privacy Model of Bohli and Pashalidis

While their system model is clearly focused on a centralised system (*i.e.* there are no distributed databases or multiple parties mentioned in their system model), Bohli and Pashalidis [72] do not explicitly state any assumptions pertaining to the underlying (centralised or distributed) structure of their privacy model. In other words, their system/privacy model may satisfy both a centralised and a distributed structure.

The privacy definitions of [72] follow a syntactic approach: as such, certain privacy notions may be dependent on the information an adversary gains access to (or is allowed to learn). Moreover, Bohli and Pashalidis consider systems that follow the non-interactive release mode. Batches of input invocations trigger a sequence of outputs, where output values represent a permutation of the input values. In particular, this permutation obfuscates the association to a user identifier — essentially, the goal, in order to release data in a privacy-preserving way. Furthermore, in their model they give considerations to stateful, stateless and online systems.

Importantly, Bohli and Pashalidis model a function that maps the serial number of the output values to the identifier of the user associated with it. By means of this function, they define privacy properties that abstract the relation between the output element and the user identifier. What is more, in [72] Bohli and Pashalidis present a hierarchy and relations among their privacy notions.

3.6 An Extension to a Semi-Distributed Privacy Model

While fully-distributed systems provide several benefits (*e.g.* (a) data remains at the data provider; (b) a reduced computation overhead[15]; and (c) there is no single point of failure) reasoning about privacy properties in such systems may be complex and challenging. One needs to take into consideration that an adversary may obtain, corrupt or control a certain threshold of system outcomes. Furthermore, privacy notions

[15] Each individual party computes the evaluation function locally on a reduced set of data (*i.e.* their own data).

need to be abstracted to include such 'collaborative' outcomes and relations between outcomes of individual systems. Thus, as a first step we consider a semi-distributed model. Data is still split over multiple parties, however users gain access to the system via a global input and view interface. As such, privacy properties may be considered only over the whole 'combined' system release. In other words, for the privacy modelling, we consider the underlying structure as a 'black box' and argue (with our notions and games) about the release of this black box. Yet, we consider data to be distributed among multiple systems (consequently, the games that abstract the privacy notions take such a distributed modelling into consideration).

3.6.1 Considerations of Individual Privacy and Group Privacy

An adversary may be able to infer knowledge about an individual by directly or indirectly targeting her (*e.g.* by grouping users according to similar properties such as their behaviour, location or user type and gaining knowledge from the observation of the group). The former case we further denote as breaching an *individual's privacy* and the latter as breaching *group privacy*. To illustrate, consider the following example: a breach of an individual's privacy would mean that an adversary is able, for any single element e_x, to infer its relation to the associated user u_x. Drawn from a genomics use case , e_x may represent a disease that user u_x has.

A breach of group privacy would mean that an adversary is able to relate two or more elements e_x, e_y, where e_x pertains to a user u_x, e_y pertains to a user u_y, and both users are member of a group $u_x, u_y \in \vec{u}_g$. In this case (assuming that e_x and e_y are types of diseases), the adversary may infer that e_x and e_y pertain to a group of similar diseases. Another example would involve a specific disease e_x. If the adversary infers that $u_x, u_y \in \vec{u}_g$, then she may learn that u_y is somehow associated to e_x. More concretely, e_x may be a specific inheritable disease and $\vec{u}_g$ represents a family, with u_x and u_y members of that family — u_y, therefore, may also be susceptible to the disease. Hence, we model our notions and games such that they give consideration to both individual and group privacy.

3.6.2 Stateful, Stateless and Online Systems

The design and application of privacy notions depends heavily on the underlying structure of a system, be it *stateful*, *stateless*, or *online*. We discuss examples following a permutation as privacy-preserving mechanism — yet any other mechanisms may apply similarly.

In a stateful system, released data may be shuffled according to a state- (or data-) dependent, potentially secret, permutation. Additionally, statistical queries may result in state-dependent answers (for example, identical queries may not be answered; result in the same answer; or have, dependent on the privacy mechanism, an increased noise value).

In a stateless system, released data may be shuffled according to a permutation chosen uniformly at random. Because stateless systems do not rely on any state (from the perspective of the system) queries submitted to it are independent and any input

has no influence on the behaviour of the system. In the design of privacy notions and privacy-preserving mechanisms we must cope with such restrictions (for example, submission of two identical queries might disclose information).

Finally, in an online system, data is processed and released individually (*i.e.* before an invocation of another input), due to its limited buffer size. Therefore, a permutation may not be applicable and other sources (such as noise addition) may need to be applied to preserve privacy in such a setting.

3.7 New Relationships and Insights between Privacy Notions

While the main contribution of Pfitzmann and Köhntop in [358] is a clear, precise yet expressive terminology of privacy notions for data minimisation, Bohli and Pashalidis' main contribution presented in [72] is a hierarchy of multiple anonymity and unlinkability variants. Our main contribution is a fusion of existing notions with the aspiration of taking the high level commonalities of each 'type' of privacy notion and the representation of these notions in a game-based fashion. In this context, this section presents new insights and relationships of the analysed privacy notions (*i.e.* those presented in [358] and [72]) and outlines foundations and principles upon which the game-based definitions arise.

3.7.1 Unobservability

In [72], Bohli and Pashalidis hinted at a notion of unobservability as a property of a system to execute *"void invocations, i.e. invocations that are not associated with anything."* Their system model, however, was too restricted to allow for such 'void' invocations. Similarly, Pfitzmann and Köhntop [358] defined a notion of unobservability (see Definition 21). For a system to be 'unobservable' it must satisfy that any adversary who is observing the systems invocations must not be able to distinguish between a 'void' invocation and a real system invocation.

In this context, we extend the framework of Bohli and Pashalidis [72] as follows. Informally stated, a system is:

- **unobservable** if an adversary is not be able to distinguish a system invocation from a 'fake' system invocation that releases a 'random' outcome element.

More formally, we state:

Definition 36 (Unobservability)**.** *A system is said to be unobservable, denoted by* UO*, if the output is indistinguishable from a random output in the range of the output parameter set, even with knowledge of the participation set* U_f*, the usage frequency set* Q_f*, and the linking relation* P_f*.*

The notion of unobservability (Def. 36) is similar to the notion of undetectability (Def. 20) proposed by Pfitzmann and Köhntopp [358]. An adversary may be able

to learn all subjects, objects and the number of objects corresponding to each subject, yet, for a single invocation of the system, she should not be able to distinguish the release outcome element from being computed over the input values or being a random element from the set of all possible outcomes.

3.7.2 Extending Bohli and Pashalidis's Hierarchy

In [72] Bohli and Pashalidis introduce relations among privacy notions via a hierarchy (see Figure 2 in [72]) that follows from the different adversarial knowledge of a notion, denoted by $K_{\mathcal{A}}$. $K_{\mathcal{A}}$, defined by Definition 39, indicates an adversary $\mathcal{A}$'s access to information about a system, ϕ (*i.e.* such as properties of a system U_f, Q_f, P_f and $|U_f|$ and information about the mapping function, f, defined by Equation 9). Bohli and Pashalidis' relations emerge as *function indistinguishability* as defined by Definition 37 (previously defined in [72] as Definition 1):

Definition 37 (Function Indistinguishability). *Within a system, ϕ, two functions, f and f', $f \neq f'$, are said, with respect to ϕ's released elements $\vec{e}$, to be:*

(a) SA-*distinguishable in any case,*
(b) PH-*distinguishable* $\iff |U_{f,\vec{e}}| = |U_{f',\vec{e}}|$,
(c) SU-*distinguishable* $\iff U_{f,\vec{e}} = U_{f',\vec{e}}$,
(d) WU-*distinguishable* $\iff Q_{f,\vec{e}} = Q_{f',\vec{e}}$,
(e) PS-*distinguishable* $\iff P_{f,\vec{e}} = P_{f',\vec{e}}$,
(f) AN-*distinguishable* $\iff P_{f,\vec{e}} = P_{f',\vec{e}}$ *and* $U_{f,\vec{e}} = U_{f',\vec{e}}$,
(g) WA-*distinguishable* $\iff P_{f,\vec{e}} = P_{f',\vec{e}}$, $U_{f,\vec{e}} = U_{f',\vec{e}}$ *and* $Q_{f,\vec{e}} = Q_{f',\vec{e}}$,
(h) UO-*distinguishable* $\iff P_{f,\vec{e}} = P_{f',\vec{e}}$, $U_{f,\vec{e}} = U_{f',\vec{e}}$ *and* $Q_{f,\vec{e}} = Q_{f',\vec{e}}$,

Our emerging privacy notions (outlined in Table 4) follow the relations defined in Definition 37 and are computationally privacy-preserving for system, ϕ, if it holds that:

Definition 38 (Computational Privacy-Preserving). *A system, ϕ, is said to be computationally privacy-preserving with respect to a privacy notion* priv $\in \{\text{SA}, \text{PH}, \text{SU}, \text{WU}, \text{PS}, \text{AN}, \text{WA}, \text{UO}\}$ *if and only if* f_{REAL} *and* f_{IDEAL} *are* priv*-distinguishable with $\vec{e} \in \mathfrak{O}$ and for all $\mathcal{A}$ it holds that:*

$$\mathbf{Adv}^{\phi}_{\mathcal{A}} = |\Pr[\mathcal{A}^{\text{priv}}_{f_{\text{REAL}}} \Rightarrow \text{true}] - \Pr[\mathcal{A}^{\text{priv}}_{f_{\text{IDEAL}}} \Rightarrow \text{true}]| \leqslant \epsilon(k)$$

with $\epsilon(k)$ a negligible function and $\mathcal{A}$'s running time is polynomial in k.

Figure 14 illustrates the relations of the privacy notions in the form of a hierarchy. The graph outlines the previous hierarchy of Bohli and Pashalidis [72] and presents the extension of our newly introduced notion of unobservability. In this context, the graph orders the notions according to the properties (or, more precisely, the outcome relation sets) an adversary is given access to learn. From the starting point on the top-left, *i.e.* the notion of strong anonymity, an adversary is able to gain more knowledge about the linking function f. As indicated by the arrows, by going from left to right in the graph $\mathcal{A}$ learns information about the user involvement. Concretely, from notion

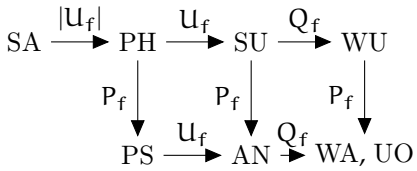


Figure 14: Hierarchy and relations between privacy notions. The graph extends the hierarchy given by Bohli and Pashalidis [72] with the notion of unobservability. From the strongest notion of strong anonymity on the top-left, an adversary $\mathcal{A}$ gains knowledge about the function f as indicated by the arrows. From left to right, $\mathcal{A}$ gains knowledge about the user involvement and from top to bottom $\mathcal{A}$ gains knowledge about the linking relation.

SA to notion PH the adversary is given access to the numbers of users that interact with the system, $|U_f|$. From the notions PH to SU and PS to AN an adversary is given access to the participants set, U_f. Likewise, the notions SU to WU and AN to WA, UO degenerate if $\mathcal{A}$ is able to learn the usage frequency set, Q_f. Finally, from top to bottom, the notions PH to PS, SU to AN and WU to AN, UO degenerate and adversary is able to learn the linking relation, P_f.

An adversary, $\mathcal{A}$, may learn said relationship sets $|U_f|, U_f, Q_f$ and P_f for any given reason[16]. For example, $\mathcal{A}$ may passively surveil users interacting with a system, ϕ, and gain access to these sets or $\mathcal{A}$ may use active mechanisms to obtain said sets. Modelling access to these sets is required for our game-based definitions and will be introduced in the next section.

3.7.3 Mapping of Adversarial Access to Information

In order to model the access of an adversary $\mathcal{A}$ to the sets $|U_f|, U_f, Q_f$ and P_f we define the following interfaces: $\mathcal{I}_{U_f}, \mathcal{I}_{Q_f}, \mathcal{I}_{P_f}$ and $\mathcal{I}_{|U_f|}$. Invocation of any of these interfaces yields the associated set representing the current system state of a system ϕ. For example, if $\mathcal{A}$ invocates $\mathcal{I}_{U_f}$ at time t_1 it would return the set U_f with all users that contributed their inputs α_{u_i} from the system start time t_0 until t_1.

A system that satisfies any of Bohli and Pashalidis' notions does not leak any other information about the linking relation f except what is provided by these interfaces. From the definitions of the privacy notions (see Section 3.2) and the previously presented hierarchy, the knowledge of an adversary, $K_{\mathcal{A}}(\cdot)$, can be abstracted by the access to the interfaces for each notion.

Definition 39 (Interface Knowledge of $\mathcal{A}$)**.** *The interface knowledge of an adversary, $\mathcal{A}$, denoted as $K_{\mathcal{A}}(\cdot)$, indicates the access to certain interfaces $\mathcal{I} \in \{\mathcal{I}_{U_f}, \mathcal{I}_{Q_f}, \mathcal{I}_{P_f}, \mathcal{I}_{|U_f|}\}$ for a privacy notion $\mathsf{priv} \in \{\mathrm{SA}, \mathrm{PH}, \mathrm{SU}, \mathrm{WU}, \mathrm{PS}, \mathrm{AN}, \mathrm{WA}, \mathrm{UO}\}$ as defined in Table 4.*

From another viewpoint, in Figure 15 we outline the relationship between our system model and Bohli and Pashalidis' notions (and our extension of unobservability).

[16] The specific reason how an adversary is able to gain knowledge of said sets is out of scope of this dissertation. We are interested in what the adversary can do in case she gains this access and model relationships how a system's ϕ properties degenerate by allowing $\mathcal{A}$ access to these sets.

[17] For the notion of PH, $\mathcal{A}$ has additional access to an oracle giving the size of the participation set $|\mathbf{U_f}|$.

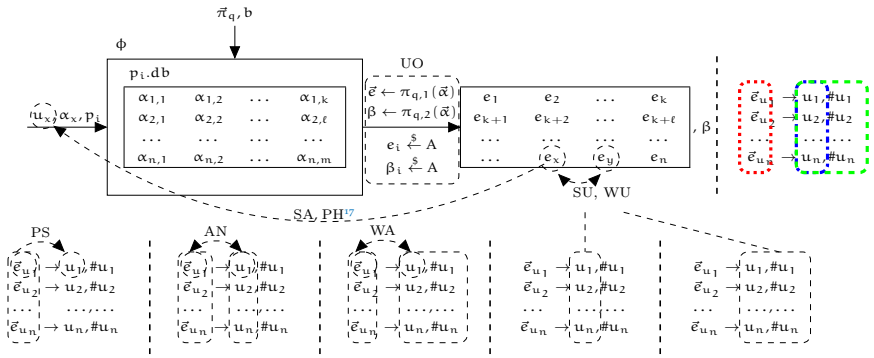


Figure 15: Illustration of privacy notions, given access to the system ϕ, and access to interfaces of the sets $\mathbf{U_f}$ (- - -), $\mathbf{Q_f}$ (- - -) and $\mathbf{P_f}$ (- - -).

There, as depicted, in dashed circles, are the sets that the adversary $\mathcal{A}$ is given access to for each notion. In each game we allow an adversary $\mathcal{A}$ to gain access to certain interfaces (*i.e.* $\mathcal{I}_{U_f}, \mathcal{I}_{Q_f}, \mathcal{I}_{P_f}$ or $\mathcal{I}_{|U_f|}$). From this access and knowledge of the outcome values $(\vec{e}, \beta)$, $\mathcal{A}$ infers knowledge about the function f. From the most restricted notion of strong anonymity, each subsequent notion obtains more information about f (via access to multiple interfaces).

The top-left part of the illustration shows a system's interaction with a user u_x who inputs a value α_x to a computational party p_i. The system ϕ is a conglomeration of multiple parties p_i and their databases $p_i.db$ store the input values α_i. The right-hand side illustrates the output of ϕ in the form of the vector $\vec{e}, \beta$.

For this concrete example, the interface $\mathcal{I}_{U_f}$ would yield the elements circled in blue; interface $\mathcal{I}_{Q_f}$ would yield the elements circled in green; and interface $\mathcal{I}_{P_f}$ would yield the elements circled in red. Not depicted, the interface $\mathcal{I}_{|U_f|}$ would yield the number of participants n.

3.7.4 Single, Multiple and Element Groups

In Table 4 we provide an overview of the analysed privacy notions by Bohli and Pashalidis. The table presents the notions via four types of categorisations; these categorisations go hand in hand and naturally fuse to the given representation:

- **Hierarchy of Bohli and Pashalidis [72]**: the table orders the privacy notions according to Bohli and Pashalidis' partial order with regards to the knowledge of an adversary from left-right and top-down.
- **Groups of notions with similar objectives**: dependent on the learning goal of a privacy notion, the notions can be categorised into: (a) single elements, (b) multiple elements, (c) element groups and (d) system behaviour.
- **Learning goal**: the learning goal of a notion indicates the objective that an adversary, $\mathcal{A}$, is aiming to learn. In other words, a privacy-preserving system aims to obfuscate such a matching in order to preserve a system's privacy property (*e.g.* strong anonymity).

Table 4: Summary of the notions for privacy considered in our privacy model.

Privacy Notion	Category⋆	Learning Goal⋆⋆	Knowledge of $\mathcal{A}^{\dagger}$
SA	single elements	$f(e_x) = u_x$	–
PH		$f(e_x) = u_x$	$\lvert U_f \rvert$
SU	multiple elements	$f(e_x) = f(e_y)$	U_f
WU		$f(e_x) = f(e_y)$	U_f, Q_f
PS	element groups	$f(\vec{e}_{u_x}) = u_x$	P_f
AN		$f(\vec{e}_{u_x}) = u_x$	U_f, P_f
WA		$f(\vec{e}_{u_x}) = u_x$	U_f, Q_f, P_f
UO	system behaviour	validity of $\vec{e}, \beta$	U_f, Q_f, P_f

⋆ The notions are grouped regarding similar objectives in distinct categories.

⋆⋆ The learning goal indicates what an adversary is aiming to achieve for each category.

† The knowledge of $\mathcal{A}$ indicates access to certain interfaces of system ϕ depending on the privacy notion under question.

◇ **Knowledge of** $\mathcal{A}$: the privacy notions can be categorised according to the knowledge of an adversary, $\mathcal{A}$, who obtains this information via access to previously mentioned interfaces.

Following the fusion of these categorisation into the representation of Table 4 the privacy notions are categorised into groups as follows: (SA, PH), (SU, WU), (PS, AN, WA) and (UO). These groups allow us to formulate informal 'reductions'[18] between members of the same group.

Definition 40 (Learning Goal)**.** *The learning goal,* $\mathtt{lern}(\cdot)$*, of a privacy notion,* $\mathtt{priv}$*, is the objective that an adversary,* $\mathcal{A}$*, aims to achieve.*

The learning goal of Bohli and Pashalidis' [72] privacy notions, *i.e.* $\mathtt{priv} \in \{\text{SA}, \text{PH}, \text{SU}, \text{WU}, \text{PS}, \text{AN}, \text{WA}, \text{UO}\}$, as defined in Definition 40, is indicated in Table 4. The learning goal is the objective that an adversary aims to learn. In other words, the learning goal is satisfied if the adversary is able to assert the learning assertion with non-negligible probability.

Definition 41 (Privacy Category)**.** *A privacy category,* $\mathtt{cat}$*, indicates certain properties of a privacy notion,* $\mathtt{priv}$*.*

In Table 4, we categorised Bohli and Pashalidis' [72] privacy notions according to the following privacy categories: (a) single elements; (b) multiple element; (c) element groups; and (d) system behaviour. Our categorisation is informed by the learning goal, $\mathtt{learn}(\mathtt{priv})$, of the privacy notions $\mathtt{priv} \in \{\text{SA}, \text{PH}, \text{SU}, \text{WU}, \text{PS}, \text{AN}, \text{WA}, \text{UO}\}$: a category is defined by the number of outcome elements $e_x \in \vec{e}$ and the number of

[18] We conjecture that such reductions hold within the presented groups since the learning goals within a group remains the same. Moreover, we present a heuristic proof following our restricted learning goal definition and association to each group's category. A deeper exploration under a fixed system model (*i.e.* defined transformation functions $\vec{\pi}_q$) is out of scope of this dissertation.

participants $u_x \in \mathcal{U}_f$ that are involved for a privacy notion $\mathtt{priv}$. In this context, the 'property' of the privacy notion is defined by the elements an adversary, $\mathcal{A}$ is required to correctly associate in order to successfully satisfy the learning assertion, $\mathtt{lern}(\mathtt{priv})$.

Definition 42 (Group of Notions)**.** *A group of privacy notions,* $\mathsf{G}_{\mathtt{priv}}$*, is a non-empty set of privacy notions belonging to the same category,* $\mathtt{cat}$*, as indicated in Table 4.*

Lemma 1 (Equivalence of Group Learning Goals)**.** *The learning goal,* $\mathtt{lern}(\mathsf{G}_{\mathtt{priv}})$*, for all privacy notions within a privacy group,* $\mathtt{priv} \in \mathsf{G}_{\mathtt{priv}}$ *is the same, i.e.* $\{\mathtt{lern}(\mathsf{G}_{\mathtt{priv}}) = \mathtt{lern}(\mathtt{priv}) : \forall \mathtt{priv} \in \mathsf{G}_{\mathtt{priv}}\}$

Proof. The proof is trivial and follows from the definitions in Table 4. □

A group of privacy notions, as defined in Definition 42, satisfies that all privacy notions, $\mathtt{priv} \in \mathsf{G}_{\mathtt{priv}}$ have the same learning goal as shown by Lemma 1.

Definition 43 (Weaker Notion (within $\mathsf{G}_{\mathtt{priv}}$))**.** *A privacy notion,* $\mathtt{priv}$*, is called 'weaker', i.e. 'weaker' secure, within a group of privacy notions* $\mathsf{G}_{\mathtt{priv}}$ *if there exists a privacy notion* $\mathtt{priv}'$ *within the same group such that for* $\mathtt{priv}'$ *an adversary* $\mathcal{A}$ *has access to a larger subset of interfaces that provide access to the sets* $\mathcal{U}_f$*,* Q_f*,* P_f *and* $|\mathcal{U}_f|$*.*

Definition 44 (Stronger Notion (within $\mathsf{G}_{\mathtt{priv}}$))**.** *A privacy notion* $\mathtt{priv}$*, is called 'stronger', i.e. 'stronger' secure, within a group of privacy notions* $\mathsf{G}_{\mathtt{priv}}$ *if there exists a privacy notion* $\mathtt{priv}'$ *within the same group such that for* $\mathtt{priv}'$ *an adversary* $\mathcal{A}$ *has access to a smaller subset of interfaces that provide access to the sets* $\mathcal{U}_f$*,* Q_f*,* P_f *and* $|\mathcal{U}_f|$*.*

Privacy notions within a privacy group $\mathsf{G}_{\mathtt{priv}}$ can be 'weaker', *i.e.* Definition 43, or 'stronger', *i.e.* Definition 44, with respect to their access to a system's ϕ interfaces. For example, in the privacy group *(b) multiple elements*, SU is the 'stronger' notion while WU is the weaker notion. The strength of a privacy notion within a group is defined with respect to the ascending order presented in Table 4.

Proposition 1. *Assume a system* ϕ*. If* ϕ *is secure under a 'weaker' privacy notion* $\mathtt{priv} \in \{\mathrm{SA}, \mathrm{PH}, \mathrm{SU}, \mathrm{WU}, \mathrm{PS}, \mathrm{AN}, \mathrm{WA}, \mathrm{UO}\}$ *within a privacy group* $\mathsf{G}_{\mathtt{priv}}$*, then* ϕ *is also secure under a 'stronger' privacy notion* $\mathtt{priv}' \in \{\mathrm{SA}, \mathrm{PH}, \mathrm{SU}, \mathrm{WU}, \mathrm{PS}, \mathrm{AN}, \mathrm{WA}, \mathrm{UO}\}$ *within the same group* $\mathsf{G}_{\mathtt{priv}}$*.*

Heuristic Proof. We present a proof by example. Assume a system ϕ is secure under the notion of PH. Applying Proposition 1 implies that ϕ is also secure under SA.

From Lemma 1 we know that the learning goal of an adversary, $\mathcal{A}$, within the group, $\mathsf{G}_{\mathtt{priv}}$ remains the same. To be secure for ϕ means, that $\mathcal{A}$ is not able to find elements within the release (*i.e.* $\vec{e}$) to satisfy the assertion stated in the group's learning goal $\mathtt{lern}$.

Given that $\mathcal{A}$ is not able to satisfy this assertion for the weaker notion $\mathtt{priv}$, where $\mathcal{A}$ has access to (an) additional interface(s), $\mathcal{A}$ does not have access to such interface(s) under the stronger notion, $\mathtt{priv}'$. As a consequence, by restricting access to the interface(s) $\mathcal{A}$ is not more likely to satisfy the group's learning goal assertion since the amount of knowledge of $\mathcal{A}$ is restricted (*i.e.* lesser than under the weak notion).

Following our example, on the one hand, for PH $\mathcal{A}$ is given access to $\mathcal{I}_{|\mathcal{U}_f|}$. On the other hand, for SA she wouldn't get access to this interface. Consequently, under PH $\mathcal{A}$ would get access to n, the number of participants and under SA she would not.

Assuming that ϕ is secure under PH implies that ϕ is secure under SA, since for both notions $\mathcal{A}$ must satisfy the same learning goal, $\mathtt{learn}$, once with additional information (which is assumed to be secure) and once without that information. □

3.7.5 Individual Privacy, Group Privacy and Semi-Distributed Databases

Pfitzmann and Köhntop's terminology [358] and Bohli and Pashalidis's privacy notions [72] are defined with individual privacy in mind. Individual privacy, as introduced in Section 3.6.1, can be formally stated within our privacy model as follows:

Definition 45 (Individual Privacy). *A system, ϕ, is secure with respect to individual privacy if it holds for a privacy notion $\mathtt{priv} \in \{\mathrm{SA}, \mathrm{PH}, \mathrm{SU}, \mathrm{WU}, \mathrm{PS}, \mathrm{AN}, \mathrm{WA}, \mathrm{UO}\}$ that a user, u_x, is fixed and an adversary, $\mathcal{A}$, must satisfy the learning goal, $\mathtt{learn}$, with non-negligible probability by determining the released elements $e_{x_1}, e_{x_2}, \ldots, e_{x_n} \in \vec{e}_{u_x} \Leftrightarrow \forall e_{x_i} \in \vec{e}_{u_x} : f(e_{x_i}) = u_x$ from all released values of ϕ, $\vec{e}_{u_x} \subseteq \vec{e}$.*

Further group privacy modifies the learning goal, $\mathtt{learn}$, of a privacy notion such that the adversary can find release elements $\vec{e}_i$ that belong to users within a group (*e.g.* $u_x, u_y \in \vec{u}_g$):

Definition 46 (Group Privacy). *A system, ϕ_g, is secure with respect to group privacy if it holds for a privacy notion $\mathtt{priv}_g \in \{\mathrm{SA}, \mathrm{PH}, \mathrm{SU}, \mathrm{WU}, \mathrm{PS}, \mathrm{AN}, \mathrm{WA}, \mathrm{UO}\}$ that a group of users, $u_x, u_y, \ldots, u_z \in \vec{u}_g$, is fixed and an adversary, $\mathcal{A}$, must satisfy the learning goal, $\mathtt{learn}$, with non-negligible probability by determining the released elements $e_x, e_y, \ldots, e_z \in \vec{e}_{u_g} \Leftrightarrow \forall e_{x_i} \in \vec{e}_{u_g} : f(e_{x_i}) = u_x, u_x \in \vec{u}_g$ from all released values of ϕ, $\vec{e}_{u_g} \subseteq \vec{e}$.*

We conjecture that the relations between the privacy notions as stated by Bohli and Pashalidis [72] (see the hierarchy presented in Figure 14) hold in both settings: (a) individual privacy and (b) group privacy.

Theorem 1 (Individually Private Systems are Computationally Privacy-Preserving). *A system, ϕ, that is secure with respect to individual privacy is computationally privacy-preserving.*

Proof. The proof is trivial and follows from our system model and our definitions (*i.e.* Definition 45 of individual privacy and Definition 38 of computationally privacy-preserving systems). □

Conjecture 1 (Group Private Systems are Computationally Privacy-Preserving). *A system, ϕ, that is secure with respect to group privacy is computationally privacy-preserving.*

Sketch Proof. Consider a system, ϕ, with privacy notion, $\mathtt{priv} \in \{\mathrm{SA}, \mathrm{PH}, \mathrm{SU}, \mathrm{WU}, \mathrm{PS}, \mathrm{AN}, \mathrm{WA}, \mathrm{UO}\}$, that is secure with respect to individual privacy and a system, ϕ_g, with privacy notion, $\mathtt{priv}_g \in \{\mathrm{SA}, \mathrm{PH}, \mathrm{SU}, \mathrm{WU}, \mathrm{PS}, \mathrm{AN}, \mathrm{WA}, \mathrm{UO}\}$, that is secure with respect to group privacy. System ϕ_g is computationally privacy-preserving if two functions f_{REAL} and f_{IDEAL} are $\mathtt{priv}_g$-distinguishable with respect to the Definition 37.

Considering the same inputs for both systems ϕ and ϕ_g the sets U_f, Q_f, P_f and $|U_f|$ differ as follows:

- $\mathbf{U_f} = \mathbf{U_{f_g}}$: the same users contributed to both systems

- $\mathbf{Q_f \neq Q_{f_g}}$: for all users $u_i \in \vec{u}_g \subset U_{f_g}$, the value $\#u_i$ of Q_f is extended to the number of total elements $\#e_i \in \vec{e} : f(e_i) = u_i \in \vec{u}_g$ associated to the group
- $\mathbf{P_f \neq P_{f_g}}$: for all users $u_i \in \vec{u}_g \subset U_{f_g}$ only one associated partition $\vec{e}_{\vec{u}_g}$ exists
- $|\mathbf{U_f}| = |\mathbf{U_{f_g}}|$: the same number of users contributed to both systems

These changes affect the quantities and distribution within the sets Q_f and P_f, but does not affect the relations and hierarchy outlined in Definition 37 and illustrated in Figure 14 (since it affects both instances of f_{REAL} and f_{IDEAL} respectively). □

Definition 47 (Internally Distributed System)**.** *A system, ϕ_δ, is internally distributed if any data values α_i, e_i and $0 \leqslant i \leqslant n$ are stored at separate, physically distributed parties, $p_i \in P$, and any user, $u_i in U_f$ interacts with the system via two global interfaces:*

- *an input interface,* $\text{input}(\cdot,\cdot,\cdot)$*, with parameter triples* (u_x, α_x, p_x) *and* $u_x \in U_f$*,* α_x*, an input value, and,* $p_x \in P$*, the distributed party*
- *an output interface,* $\text{view}^{\vec{\pi}_q}(\cdot)$*, with parameter* $p_x \in P$*, the distributed party*

The internal distribution of the databases of a system, as defined in Definition 47, does not influence the relations stated by Bohli and Pashalidis [72] (see the hierarchy presented in Figure 14).

Theorem 2 (Internally Distributed Systems are Computationally Privacy-Preserving)**.** *A system, ϕ_δ, that is internally distributed is computationally privacy-preserving.*

Proof. The internal distribution of the data (via parties $p_i \in P$ with $0 \leqslant i \leqslant n$) does not influence the system's global sets U_f, Q_f and P_f and the interaction with ϕ_δ is via the same global interfaces $\text{input}(\cdot,\cdot,\cdot)$ and $\text{view}^{\vec{\pi}_q}(\cdot)$ as a centralised system, ϕ. □

3.7.6 Insights for One-Element Systems

In [72], Bohli and Pashalidis showed some insights of privacy notions with respect to their operational characteristics: *i.e.* (a) stateful systems are priv-distinguishable for all privacy notions outlined in Table 4; (b) stateless systems are *always* {WU, WA}-distinguishable; and (c) online systems can not be {SU, WU, AN, WA}-distinguishable, but only {SA, PH}-distinguishable.

In the following, we present new insights with respect to a new class of one-element systems (outlined in Definition 48). Examples of one-element systems are auction systems and election systems where a user must only input exactly one element.

Definition 48 (One-Element System)**.** *A one-element system, denoted by ϕ_1, is a system where each participant u_x is only allowed to contributed exactly one input element α_x to the system's processing.*

Before presenting specific insights and characteristics of one-element systems, generally, it holds that for any system, ϕ, that adversarial knowledge of the set Q_f implies knowledge of the set U_f as shown by Lemma 2:

Lemma 2 (Knowledge of Q_f implies U_f)**.** *For any system ϕ, knowledge of the usage frequency set Q_f implies knowledge of the participant set U_f.*

Proof. The proof is trivial and follows from the definition of the sets U_f and Q_f (*i.e.* Definition 33 and Definition 34, respectively). Concretely, U_f is a subset of Q_f, whereas Q_f extends U_f by adding the number of elements e_x a participant u_x for each entry $(u_x, \#u_x)$ in Q_f contributes to ϕ. □

In addition, for one-element systems, ϕ_1, the reverse relationship also holds, *i.e.* knowledge of the set U_f implies Q_f as shown by Lemma 3:

Lemma 3 (Knowledge of U_f implies Q_f)**.** *For any one-element system ϕ_1 knowledge of the participant set U_f implies knowledge of the usage frequency set Q_f.*

Proof. The proof is straightforward and follows from the definition of a one-element system, *i.e.* Definition 48. Since each participant u_x contributes exactly one input element α_x to the system ϕ_1, Q_f can be derived from knowledge of U_f and setting the value $\#u_x$ for each entry $(u_x, \#u_x)$ in Q_f to 1. □

Moreover, for one-element systems, ϕ_1, the generic output of a system $\vec{e}$ implies knowledge of the set P_f as shown by Lemma 4:

Lemma 4 ($\vec{e}$ implies knowledge of P_f)**.** *For any one-element system ϕ_1 knowledge of the release elements $\vec{e}$ implies knowledge of the linking relation set P_f. Moreover, the released elements are equivalent to the linking relation set $\vec{e} = P_f$.*

Proof. The proof is straightforward and follows from the definition of a one-element system, *i.e.* Definition 48. Since each participant u_x contributes exactly one input element α_x to the system ϕ_1, P_f can be derived from the access of the released elements $\vec{e}$. As a consequence it holds that $\vec{e} = P_f$ following Definition 35. □

For the further characteristics of (one-element) systems, we denote two similarity definitions of privacy notions: (a) similarity and (b) strict similarity:

Definition 49 (Similarity of Privacy Notions)**.** *Two privacy notions* $\mathtt{priv}, \mathtt{priv}' \in \{\mathrm{SA}, \mathrm{PH}, \mathrm{SU}, \mathrm{WU}, \mathrm{PS}, \mathrm{AN}, \mathrm{WA}, \mathrm{UO}\}$ *are similar, denoted by the symbol* $\mathtt{priv} \cong \mathtt{priv}'$*, with respect to an adversary, $\mathcal{A}$'s, access to the notions interface knowledge* $K_{\mathcal{A}}(\cdot)$*, if* $K_{\mathcal{A}}(\mathtt{priv}) = K_{\mathcal{A}}(\mathtt{priv}')$.

Definition 50 (Strict Similarity of Privacy Notions)**.** *Two privacy notions* $\mathtt{priv}, \mathtt{priv}' \in \{\mathrm{SA}, \mathrm{PH}, \mathrm{SU}, \mathrm{WU}, \mathrm{PS}, \mathrm{AN}, \mathrm{WA}, \mathrm{UO}\}$ *are strictly similar, denoted by the symbol* $\mathtt{priv} \cong_S \mathtt{priv}'$*, with respect to an adversary, $\mathcal{A}$'s, access to the notions interface knowledge,* $K_{\mathcal{A}}(\cdot)$*, and learning goal,* $\mathtt{learn}(\cdot)$*, if* $K_{\mathcal{A}}(\mathtt{priv}) = K_{\mathcal{A}}(\mathtt{priv}')$ *and* $\mathtt{learn}(\mathtt{priv}) = \mathtt{learn}(\mathtt{priv}')$.

With the term '(strict) similarity' of two privacy notion $\mathtt{priv}, \mathtt{priv}'$ we understand the equivalence with respect to their interface access and learning goal. In this context, Theorem 3 shows similarity of the notion of WU with the notion of SU, for any one-element system, ϕ_1:

Theorem 3 ($\mathrm{WU} \cong \mathrm{SU}$)**.** *For any one-element system ϕ_1, it holds that* $\mathrm{WU} \cong_S \mathrm{SU}$.

Proof. We provide the proof in multiple steps. In order to satisfy strict similarity between WU and SU we need to show that the following conditions hold:

(a) equivalence of the interface knowledge, *i.e.* $K_{\mathcal{A}}(WU) = K_{\mathcal{A}}(SU)$

(b) equivalence of the learning goal, *i.e.* $learn(WU) = learn(SU)$

For (a), the definition of $K_{\mathcal{A}}(WU)$ shows that for WU an adversary, $\mathcal{A}$, knows the sets U_f and Q_f. Furthermore, from the definition of $K_{\mathcal{A}}(SU)$, $\mathcal{A}$ knows the set U_f. Using Lemma 3, for a one-element system, we know that U_f implies knowledge of Q_f. As such, (a) is satisfied since for $K_{\mathcal{A}}(SU)$ $\mathcal{A}$ can apply Lemma 3 to gain Q_f — consequently, $K_{\mathcal{A}}(WU) = K_{\mathcal{A}}(SU)$.

For (b), from Table 4 we observe that WU and SU belong to the same privacy group G_{priv}. Further, using Lemma 1 proves that indeed $learn(WU) = learn(SU)$. □

Moreover, Theorem 4 shows strict similarity of the notion of WA and AN:

Theorem 4 (WA $\cong_S$ AN)**.** *For any one-element system* ϕ_1*, it holds that* WA $\cong_S$ AN*.*

Proof. The proof is straightforward and follows the same arguments as the proof of Theorem 3. In order to satisfy strict similarity between WA and AN the same conditions as outlined in Theorem 3 (applied for WA and AN) must hold:

For (a), the definition of $K_{\mathcal{A}}(WA)$ shows that for WA an adversary, $\mathcal{A}$, knows the sets U_f, Q_f and P_f. Further, from the definition of $K_{\mathcal{A}}(AN)$, $\mathcal{A}$ knows the set U_f and P_f. Using Lemma 3, for a one-element system, we know that U_f implies knowledge of Q_f. As such, (a) is satisfied since for $K_{\mathcal{A}}(AN)$ $\mathcal{A}$ can apply Lemma 3 to gain Q_f — consequently, $K_{\mathcal{A}}(WA) = K_{\mathcal{A}}(AN)$.

For (b), from Table 4 we observe that WA and AN belong to the same privacy group G_{priv}. Further, using Lemma 1 proves that indeed $learn(WA) = learn(SA)$. □

3.7.7 An Informal Mapping between Notions

In this chapter we have seen, on the one hand, the privacy model of Pfitzmann and Köhntopp [358], which focuses on a textual representation to clarify the terminology and terms surrounding privacy notions. On the other hand, the privacy model of Bohli and Pashalidis [72] focuses on a formal representation to show a hierarchy and relationships between privacy notions.

With our contributions, we aim for a synergy of the previous contribution with the use of techniques of provable security and game-based privacy representations to bridge the understanding of privacy notions by privacy researchers and foster the analysis of privacy-preserving systems via privacy notions.

In order to enhance understanding of the different models, in Table 5, we present a mapping between the different privacy models. Such mapping connects and compares the privacy models with regards to the following characteristics: (a) the privacy notions of each model; (b) the entities (*i.e.* users, input and output of a system ϕ); (c) the transformation entity between 'inputs' and 'outputs' of a system ϕ; (d) the abstraction of the model; (e) the underlying security principle; (f) the privacy categories; and (g) the supported system models.

Table 5: Mapping between the privacy models of Pfitzmann and Köhntop [358], Bohli and Pashalidis [72] and ours.

Categories	**Pfitzmann & Köhntop [358]**	**Bohli & Pashalidis [72]**	**Our Model**
Privacy Notions	AN, UN*, PS, UD**, UO	SA, PH, SU, WU, PS AN, WA	SA, PH, SU, WU, PS AN, WA, UO
Entities			
Users	'subjects'	$u_i \in U_f$	$u_i \in U_f$
Inputs	'objects'	$\alpha_i \in A$	$\alpha_i \in \mathfrak{I}$
Outputs	'objects'	$i \in I, i \in P_f$	$e_i \in \mathfrak{O}, e_i \in P_f$
Transf. Function	-	nextBatch, π_j, Π_j	$\vec{\pi}_q$
Abstraction	textual	formal	game-based
Security Principle	$\mathcal{A}$'s a-priori vs. a-posteriori distinguisher	left world vs. right world distinguisher	ideal model vs. real model distinguisher
Privacy Considerations	✓ individual privacy ✗ group privacy	✓ individual privacy ✗ group privacy	✓ individual privacy ✓ group privacy
System Model	✓ centralised ✗ distributed	✓ centralised ✗ distributed	✓ centralised ✓ (semi)-distributed

* UN ... Unlinkability

** UD ... Undetectability

Table 6: Informal Mapping between the privacy notions of Pfitzmann and Köhntop [358], Bohli and Pashalidis [72] and ours, with respect to the 'objective' of a privacy notion.

Objectives	**Pfitzmann & Köhntop [358]**	**Bohli & Pashalidis [72]**	**Our Model**
Anonymity	AN	SA, AN, WA	SA, AN, WA
Unlinkability	UN	SU, WU	SU, WU
Participation Hiding	-	PH	PH
Pseudonymity	PS	PS	PS
Unobservability	UD, UO	-	UO

Moreover, in Table 6 we present a mapping between the different privacy models with respect to the 'objective' of the privacy notions of each model. In this context, the notions are loosely categorised according to the privacy 'principle' without the aim of a formal equivalence. (The groups are formed with respect to a mutual understanding of the 'objective' of a notion).

3.8 Privacy Games

In the following, we briefly state our adversarial model, then elaborate on the fundamental definitions and interactions of our privacy games and, importantly, present the abstraction of the previously defined privacy notions in the form of games.

3.8.1 Adversarial Model and Principles of Privacy Games

Adversarial Model. An adversary, denoted by $\mathcal{A}$, is represented by an algorithm that runs in polynomial time. $\mathcal{A}$ is given access to a limited number of interfaces, as defined by the privacy game. An interface, denoted by $\mathcal{I}$, gives $\mathcal{A}$ access to certain knowledge about ϕ and relations between the outcome values $(\vec{e}, \beta)$. The interaction of $\mathcal{A}$ with an interface $\mathcal{I}$ is further denoted by $\mathcal{A}^{\mathcal{I}}$. Within the games, interfaces are represented as procedures, denoted as **proc**. Each game consists of a limited number of such procedures, which outline the necessary steps in the form of an algorithm that an adversary needs to perform in order to break the associated notion.

Overall, we say that $\mathcal{A}$ is interested in learning the mapping function f, or any interesting properties about f, represented by the defined notions in Section 3.2. Our selected notions are based upon the notions of Bohli and Pashalidis [72] and modified to fit with our system model as described in Table 5 and Table 6. In the following, each of these notions are represented as a game. We define a game G, and say that an adversary, $\mathcal{A}$, 'breaks'[19] a privacy notion if she is able to come up with a 'strategy' (in the literature often referred to as a 'distinguisher') such that, for multiple executions of a privacy game, the associated game always evaluates to true with a non-negligible probability. The phrasing 'non-negligible' is here defined with respect to the size of the anonymity set (*i.e.* number of users / elements within the system ϕ). As such, we require $|\vec{e}|$ and $|U_f|$ to be large[20], as otherwise $\mathcal{A}$ would always have a non-negligible success probability. The success probability of $\mathcal{A}$ is denoted by

$$\mathbf{Succ}(\mathcal{A}) = \Pr[G_{\star}^{\mathcal{A}} \Rightarrow \text{true}] \tag{8}$$

where $\star \in \{SA, PH, SU, WU, PS, AN, WA, UO\}$[21] denotes any of the games.

Another measure, for example, the distinguishing advantage of $\mathcal{A}$, as defined by Definition 38, would require the definition of 'idealised' instances of privacy-preserving systems with regards to the aforementioned properties (*e.g.* an idealised instance with regards to unlinkability would always return unique 'unlinked' outcome elements).

In order to ensure correctness, we assume $\mathcal{A}$ to be semi-honest: when $\mathcal{A}$ 'plays' a game, we assume that it behaves truthfully and follows the game's rules. In other words, $\mathcal{A}$ does not deviate from the game. In particular, this means that $\mathcal{A}$ does not submit elements that it has not previously obtained from the system ϕ (*e.g.* it does submit a random value $e_x \notin \vec{e}$, where $\vec{e} \leftarrow \mathcal{A}^{\text{view}^{\vec{\pi}_q}()}$), or from any of the other interfaces $\mathcal{I}_{U_f}, \mathcal{I}_{Q_f}, \mathcal{I}_{P_f}$ or $\mathcal{I}_{|U_f|}$, which reveal specific information about certain subsets of f.

Precisely, $\mathcal{A}$ is allowed to interact adaptively with the interfaces of ϕ, and, at a given point in time, must commit an input value pair $(u_x, \vec{\alpha}_x)$ to which it receives the output tuple $(\vec{e}_x, \beta_x)$ from ϕ, which we further denote as *general observation*. Given

[19] A system, ϕ, achieves or gives guarantees conveyed by a privacy notion. With 'break' we mean that an adversary is able to win a game (*i.e.* the game returns true) and, as such, ϕ does not retain these guarantees anymore.

[20] By 'large' we mean, that it is computationally infeasible to perform random guesses.

[21] Here, the notation follows the notions of strong anonymity (SA), participation hiding (PH), strong unlinkability (SU), weak unlinkability (WU), pseudonymity (PS), anonymity (AN), weak anonymity (WA), and unobservability (UO).

proc$_{\circlearrowleft}$ $\mathcal{I}_{U_f}$
  forall i in range$(0, |\vec{e}|)$:
    $U_f \leftarrow U_f \cup f(e_i)$
  return U_f

Figure 16: Interface $\mathcal{I}_{U_f}$ (Participant Set).

proc$_{\circlearrowleft}$ $\mathcal{I}_{Q_f}$
  forall i in range$(0, |\vec{e}|)$:
    $Q_f \leftarrow Q_f \cup^{\dagger} \{f(e_i)\}$
  return Q_f

Figure 17: Interface $\mathcal{I}_{Q_f}$ (Usage Frequency Set).

$^{\dagger}\cup$ is here (and only here) implemented such that if $f(e_i)$ already exists in $Q_f = \{(u_i, \#u_i)\}$, then it solely increases frequency counter $\#u_i$.

proc$_{\circlearrowleft}$ $\mathcal{I}_{P_f}$
  $P_f \leftarrow \text{part}(\vec{e})$
  return P_f

Figure 18: Interface $\mathcal{I}_{P_f}$ (Linking Relation).

proc$_{\circlearrowleft}$ $\mathcal{I}_{|U_f|}$
  $|U_f| \leftarrow \text{sizeof}(\mathcal{A}^{\mathcal{I}_{U_f}})$
  return $|U_f|$

Figure 19: Interface $\mathcal{I}_{|U_f|}$ (Number of Participants).

these parameters (and access to some system-specific interfaces, as defined by the games in the following) it will be evaluated if $\mathcal{A}$ is able to successfully break the given notion. Further, for individual privacy, we assume u_x to be fixed; for group privacy, we assume $u_x \in \vec{u}_g$, where $\vec{u}_g$ is a fixed group of user identifiers.

These restrictions allow us to abstract (and in some cases omit) some of ϕ's specific operations and tasks in order to keep the following games concise and clearly readable. To this end, we do not include checks of validity (for example, $u_x \in U_f$, for input or validate procedures) for any of the values submitted to the interfaces.

Definition of Interfaces. Depending on the privacy game, an adversary, $\mathcal{A}$, is given access to a limited number of interfaces of system ϕ. Figures 16-19 present the definitions of the interfaces $\mathcal{I}_{U_f}, \mathcal{I}_{Q_f}, \mathcal{I}_{P_f}$ and $\mathcal{I}_{|U_f|}$. More concretely, $\mathcal{I}_{U_f}$ represents the interface for the participant set. As such, it returns a set of all user identifiers u_i. $\mathcal{I}_{Q_f}$ represents the interface for the usage frequency set. It returns a set of user identifiers u_i and number of elements associated to the user identifier $\#u_i$. $\mathcal{I}_{P_f}$ represents the interface for the linking relation. It returns a set of outcome elements partitioned by user identifiers u_is. Finally, $\mathcal{I}_{|U_f|}$ represents the interface for the number of participants. It returns the number of users which participated in the system.

In the representation of the interfaces, we omit any concrete implementation details. Here, we only aim to define to what these interfaces are providing access to, rather than how such an interface is implemented in a real-world system.

Principles of Privacy Games. Any game $G_\star$ consists of a limited number of procedures. A procedure, denoted by **proc**, defines the computational steps an algorithm has to follow in order to terminate. Procedures are executed top-down and in order. Further, some procedures may only be executed once during an attack session, and others may be executed an arbitrary number of times, indicated by **proc**$_{\circlearrowleft}$. This allows $\mathcal{A}$ to 'play' around a bit and get used to the system, before, for a fixed set of

proc initialise
forall i in range$(0, n)$:
$p_i.db \leftarrow \emptyset$

proc$_{\circlearrowleft}$ input(u_i, α_i, p_i)
$p_i.db \leftarrow p_i.db \cup (u_i, \alpha_i)$

proc$_{\circlearrowleft}$ corrupt(p_i)
return $p_i.db$

proc$_{\circlearrowleft}$ view$^{\vec{\pi}_q}$
forall i in range$(0, n)$:
$(\vec{e}_i, \beta_i) \leftarrow \pi_{q,i}(p_i.db)$
$\vec{e} \leftarrow \vec{e}_1 \circ \vec{e}_2 \circ \cdots \circ \vec{e}_n$
$\beta \leftarrow \beta_1 \circ \beta_2 \circ \cdots \circ \beta_n$
return $(\vec{e}, \beta)$

Figure 20: Game G.

inputs, she tries to break the privacy notion, by executing **proc** $\star$, where $\star \in \{\text{SA}, \text{PH}, \text{SU}, \text{WU}, \text{PS}, \text{AN}, \text{WA}, \text{UO}\}$.

Importantly, an adversary may only be able to query each of the various validate interfaces once. Thereafter, the dataset needs to be destroyed[22] to maintain a guarantee of privacy — to prevent $\mathcal{A}$ from submitting, in a brute-force fashion, queries to the validate interface in the hope of hitting the correct result. In real systems this may not always be applicable. As such, in a more restricted setting it can be argued that privacy may only be guaranteed as long as an adversary is not exceeding a previously defined number of queries to the validate interface.

In the following, we define a basic game G, which provides procedures used by all games $G_\star$. We define *inheritance* as follows:

Definition 51 (Inheritance)**.** *A game* G_x *inherits all procedures of game* G_y*, denoted* $G_x \vDash G_y$*, and* G_x *can, additionally to its own procedures, call any of the procedures of* G_y*.*

In our general case: $G_\star \vDash G$, *i.e.* the games $G_\star$ inherit all procedures from the basic game G. In Figure 20, we present game G. $\mathcal{A}$ may play game G for an *arbitrary* number of times, before it plays a derived game $G_\star$ *once*, where she aims to break the expressed notion.

We denote with $a \leftarrow x$ an assignment (of value x to element a), $a \xleftarrow{\$} A$ denotes a uniform random assignment of a value of set A to element a, $\emptyset$ denotes the empty set, $A \cup B$ denotes the union of sets A and B, and $A \cup \{b\}$ denotes the union of set A and $\{b\}$, with b being a single element in a set.

In general, $\mathcal{A}$ interacts with ϕ ('plays' the games) in the following way:

1. $\mathcal{A}$ inputs data (α_i) into the system ϕ.
2. $\mathcal{A}$ obtains[23] output $(\vec{e}, \beta)$ from the system ϕ.
3. (optionally) $\mathcal{A}$ may gain additional information by corrupting a threshold of parties.
4. $\mathcal{A}$ selects any value from the output, to which it believes it knows the relation (specified by the privacy notions) and submits that to the validate interface (*e.g.* any $e_x \in \vec{e}_x$, where $\mathcal{A}$ believes to know u_x).
5. Game $G_\star$ returns true or false according to the output of the validate interface.

[22] By 'destroyed' we mean that the dataset must not be used anymore, as otherwise it would leak information about its contents.

[23] $\mathcal{A}$ obtains $(\vec{e}, \beta)$ by querying the view interface. $\vec{e}_x \circ \vec{e}_y$, for example, denotes the composition of two sets $\vec{e}_x$ and $\vec{e}_y$. This means that $\pi_{q,i}(\vec{e}_x) \circ \pi_{q,i}(\vec{e}_y)$ is equivalent to $\pi_{q,i}(\vec{e}_x \cup \vec{e}_y)$. Similarly, this also holds for β.

proc validate$_{SA}(e_x, u_x)$
if $f(e_x) = u_x$ then: return true
else: return false

proc SA
$\mathcal{A}^{input(u_x, \alpha_x, p_x)}$
$\vec{e}_x, \beta_x \leftarrow \mathcal{A}^{view^{\vec{\pi}_q}()}$
return $\mathcal{A}^{validate_{SA}(e_x, u_x)}$

Figure 21: Game G_{SA} (Strong Anonymity).

proc$_{\circlearrowleft}$ $\mathcal{I}_{|U_f|}$
return $|U_f|$

proc validate$_{PH}(e_x, u_x)$
if $f(e_x) = u_x$ then: return true
else: return false

proc PH
$\mathcal{A}^{input(u_x, \alpha_x, p_x)}$
$\vec{e}_x, \beta_x \leftarrow \mathcal{A}^{view^{\vec{\pi}_q}()}$
return $\mathcal{A}^{validate_{PH}(e_x, u_x)}$

Figure 22: Game G_{PH} (Participation Hiding).

3.8.2 Modelling Privacy Notions as Privacy Games

In the following, we present the privacy games that we consider in our privacy model. In detail, for each of the selected privacy notions, we formulate a game that an adversary has to win in order to break the associated notion. As mentioned previously, the privacy notions follow and extend the work of Bohli and Pashalidis [72]; the game-based definitions are novel. In formulating said games, we take into account that they must abstract the behaviour of a system (*i.e.* it may be stateful, stateless or online), must abstract the locality of the computational parties (*i.e.* centralised or distributed), and must give consideration to individuals as well as groups.

In Figures 21-28, we illustrate the games $G_\star$, with $\star \in \{SA, PH, SU, WU, PS, AN, WA, UO\}$. Most of the games have similarities (*e.g.* they differ only in the validate interface or differ by the access to interfaces $\mathcal{I}_{U_f}, \mathcal{I}_{Q_f}, \mathcal{I}_{P_f}$ or $\mathcal{I}_{|U_f|}$). Thus, we group certain games and notions with similar objectives as follows: (SA, PH), (SU, WU), (PS, AN, WA) and (UO).

Privacy Games for Single Elements. In Figure 21, we present game G_{SA} (strong anonymity) and, in Figure 22, game G_{PH} (participation hiding), respectively. The objective of group (SA, PH) is, from the outcome $\vec{e}_x, \beta_x$, to find any value $e_x \in \vec{e}_x$ for which $\mathcal{A}$ is able to identify the user identifier u_x. The validate functions of this group return true if $\mathcal{A}$ is able to do so.

A system ϕ that satisfies the property of (SA, PH) prevents $\mathcal{A}$ from relating output elements to their associated user identifiers. More precisely, $\mathcal{A}$ must not be able, for any element $e_x \in \vec{e}_x$, to learn the associated identifier u_x, such that $f(e_x) = u_x$. In the case of SA, $\mathcal{A}$ is not given anything beyond the general observation. In the case of PH, in addition to the general observation, we assume that $\mathcal{A}$ is able to learn $|U_f|$, which might or might not help to find said relation.

Privacy Games for Multiple Elements. In Figure 23, we present game G_{SU} (strong unlinkability) and, in Figure 24, game G_{WU} (weak unlinkability), respectively. The objective of group (SU, WU) is, from the outcome $\vec{e}_x, \beta_x$, to find any two values

proc$_{\circlearrowleft}$ $\mathcal{I}_{U_f}$
  return U_f

proc validate$_{SU}(e_x, e_y, u_x)$
  if $f(e_x) = f(e_y) = u_x$ then: return true
  else: return false

proc SU
  $\mathcal{A}^{\text{input}(u_x,\alpha_x,p_x)}$
  $\mathcal{A}^{\text{input}(u_x,\alpha_y,p_x)}$
  $\vec{e}_x, \beta_x \leftarrow \mathcal{A}^{\text{view}^{\vec{\pi}_q}()}$
  return $\mathcal{A}^{\text{validate}_{SU}(e_x,e_y,u_x)}$

Figure 23: Game G_{SU} (Strong Unlinkability).

proc$_{\circlearrowleft}$ $\mathcal{I}_{U_f}$
  return U_f

proc$_{\circlearrowleft}$ $\mathcal{I}_{Q_f}$
  return Q_f

proc validate$_{WU}(e_x, e_y, u_x)$
  if $f(e_x) = f(e_y) = u_x$ then: return true
  else: return false

proc WU
  $\mathcal{A}^{\text{input}(u_x,\alpha_x,p_x)}$
  $\mathcal{A}^{\text{input}(u_x,\alpha_y,p_x)}$
  $\vec{e}_x, \beta_x \leftarrow \mathcal{A}^{\text{view}^{\vec{\pi}_q}()}$
  return $\mathcal{A}^{\text{validate}_{WU}(e_x,e_y,u_x)}$

Figure 24: Game G_{WU} (Weak Unlinkability).

$e_x, e_y \in \vec{e}_x$ for which $\mathcal{A}$ believes they are related to the user identifier u_x. The validate functions of this group return true if $\mathcal{A}$ is able to do so.

A system ϕ that satisfies the property of (SU, WU) prevents $\mathcal{A}$ from relating output elements that pertain to the same user identifier. More precisely, $\mathcal{A}$ must not be able, for a fixed user identifier u_x, to find any two elements $e_x, e_y \in \vec{e}_x$, such that $f(e_x) = f(e_y) = u_x$. In the case of SU, in addition to the general observation, we assume that $\mathcal{A}$ is able to learn U_f, which might or might not help to find said relation. In the case of WU, in addition to the general observation, we assume that $\mathcal{A}$ is able to learn U_f and Q_f. This allows $\mathcal{A}$ to learn, for each user, the number of elements of its outcome set, which might or might not help to find said relation.

Privacy Games for Element Groups. In Figure 25, we present game G_{PS} (pseudonmyity), in Figure 26, game G_{AN} (anonymity) and, in Figure 27, game G_{WA} (weak anonymity), respectively. The objective of group (PS, AN, WA) is, from the outcome $\vec{e}_x, \beta_x$, given the ability to group certain outcome elements together, for this very group $\vec{e}_{u_x}$, to identify the user identifier u_x. The validate functions of this group return true if $\mathcal{A}$ is able to do so.

A system ϕ that satisfies the property of (PS, AN, WA) prevents $\mathcal{A}$ from relating a group of output elements (or an element within that group), which pertain to the same user identifier, to the user identifier. More precisely, $\mathcal{A}$ must not be able, for a fixed user identifier u_x, to find the group of related outcome elements $\vec{e}_{u_x} \subseteq \vec{e}_x$ such that $\forall e_x \in \vec{e}_{u_x} | f(e_x) = u_x$. In the case of PS, in addition to the general observation, we assume that $\mathcal{A}$ is able to learn P_f. This allows $\mathcal{A}$ to assign a pseudonym to each partition of the outcome set $\vec{e}_x$, which might or might not help to find said relation. In the case of AN, in addition to the general observation, we assume that $\mathcal{A}$ is able to learn U_f and P_f. This allows $\mathcal{A}$ to learn the user identifiers associated with the outcome values, and the ability to learn partitions of the overall outcome set $\vec{e}_x$, but not the relationship between those sets, which might or might not help to find said relation.

proc$_{\circlearrowleft}$ $\mathcal{I}_{\mathbf{P_f}}$
  return P_f

proc validate$_{PS}(\vec{e}_{u_x}, u_x)$
  if $f(\vec{e}_{u_x}) = u_x$ then: return true
  else: return false

proc PS
  $\mathcal{A}^{input(u_x,\alpha_x,p_x)}$
  $\vec{e}_x, \beta_x \leftarrow \mathcal{A}^{view^{\vec{\pi}_q}()}$
  return $\mathcal{A}^{validate_{PS}(\vec{e}_{u_x},u_x)}$

Figure 25: Game G_{PS} (Pseudonymity).

proc$_{\circlearrowleft}$ $\mathcal{I}_{\mathbf{U_f}}$
  return U_f

proc$_{\circlearrowleft}$ $\mathcal{I}_{\mathbf{P_f}}$
  return P_f

proc validate$_{AN}(\vec{e}_{u_x}, u_x)$
  if $f(\vec{e}_{u_x}) = u_x$ then: return true
  else: return false

proc AN
  $\mathcal{A}^{input(u_x,\alpha_x,p_x)}$
  $\vec{e}_x, \beta_x \leftarrow \mathcal{A}^{view^{\vec{\pi}_q}()}$
  return $\mathcal{A}^{validate_{AN}(\vec{e}_{u_x},u_x)}$

Figure 26: Game G_{AN} (Anonymity).

Finally, in the case of WA, in addition to the general observation, we assume that $\mathcal{A}$ is able to learn U_f, Q_f and P_f[24]. This allows $\mathcal{A}$ to learn the user identifiers associated with the outcome values, partitions of the overall outcome set $\vec{e}_x$ and information about the number of elements each user has contributed in the outcome set, which might or might not help to find said relation.

Privacy Games for System Behaviour. We extend the privacy notions of Bohli and Pashalidis [72] by the notion of *unobservability* (see Def. 36). In Figure 28, we present the associated game G_{UO} (unobservability). The objective of group (UO) is, from the outcome $\vec{e}_x, \beta_x$, to distinguish if the values $\vec{e}_{u_x}, \beta_{u_x}$ were generated as a result of a system invocation, or generated purely at random. The validate functions of this group return true, if $\mathcal{A}$ is able to do so.

A system ϕ that satisfies the property of UO prevents $\mathcal{A}$ from distinguishing that, for a fixed tuple (u_x, α_x) of user identifier and input value, a system invocation has taken place or not. Hereby, we mean that $\mathcal{A}$ is able to observe the overall system outcome $\vec{e}_x$, where all elements $\vec{e}_x \backslash \{e_x\}$ are result of a system invocation of $\vec{e}_x \leftarrow \pi_{q,1}(\vec{\alpha}_x)$ except for the fixed input value α_x. To clarify, in other words, with system invocation we understand that ϕ outputs a single element e_x. In this context, it may be easy to distinguish a system ϕ behaving in the above way, by simply determining $|\vec{e}_x|$. If $\alpha_x \notin \vec{\alpha}_x$, then $|\vec{e}_x|_{\alpha_x \notin \vec{\alpha}_x} < |\vec{e}_x|_{\alpha_x \in \vec{\alpha}_x}$, where in the latter case $\alpha_x \in \vec{\alpha}_x$. Therefore, even for a *void* invocation of ϕ, it must hold that $\exists e_x \in \vec{e}_x | e_x \not\leftarrow \pi_{q,1}(\alpha_x)$. The value e_x must be produced in a way that is indistinguishable of a system being invoked or not (in our game-based definition we say the outcome element gets assigned to a random value from the parameter space, *i.e.* $e_x \xleftarrow{\$} \mathfrak{O}$). Consequently, to achieve

[24] In the case that each user contributes a unique amount of elements to ϕ, and $\mathcal{A}$ is able to infer this knowledge, then it is easy to see that $\mathcal{A}$ is able to determine $f(e_x) = u_x$ by the mere combination of the sets U_f, Q_f and P_f. However, in other cases, where this condition is not satisfied (for example, the number of elements each user contributes is uniformly distributed), $\mathcal{A}$ only partially gains information by the combination of the previously mentioned sets.

$\underline{\textbf{proc}_{\circlearrowleft}\ \mathcal{I}_{\mathbf{U_f}}}$
return U_f

$\underline{\textbf{proc}_{\circlearrowleft}\ \mathcal{I}_{\mathbf{Q_f}}}$
return Q_f

$\underline{\textbf{proc}_{\circlearrowleft}\ \mathcal{I}_{\mathbf{P_f}}}$
return P_f

$\underline{\textbf{proc validate}_{WA}(\vec{e}_{u_x}, u_x)}$
if $f(\vec{e}_{u_x}) = u_x$ then: return true
else: return false

$\underline{\textbf{proc WA}}$
$\mathcal{A}^{\text{input}(u_x, \alpha_x, p_x)}$
$\vec{e}_x, \beta_x \leftarrow \mathcal{A}^{\text{view}^{\vec{\pi}_q}()}$
return $\mathcal{A}^{\text{validate}_{WA}(\vec{e}_{u_x}, u_x)}$

Figure 27: Game G_{WA} (Weak Anonymity).

$\underline{\textbf{proc}_{\circlearrowleft}\ \mathcal{I}_{\mathbf{U_f}}}$
return U_f

$\underline{\textbf{proc}_{\circlearrowleft}\ \mathcal{I}_{\mathbf{Q_f}}}$
return Q_f

$\underline{\textbf{proc}_{\circlearrowleft}\ \mathcal{I}_{\mathbf{P_f}}}$
return P_f

$\underline{\textbf{proc validate}_{UO}(g)}$
return $(g == b)$

$\underline{\textbf{proc}_{\circlearrowleft}\ \textbf{view}^{\vec{\pi}_q}_{\textbf{refut.}}}$
$b \xleftarrow{\$} \{0, 1\}$
forall i in range$(0, n)$:
$\vec{e}_i \leftarrow \pi_{q,1}(p_i.db)$
$\beta_i \leftarrow \pi_{q,2}(pi.db)$
if $b = 0$ then: $\vec{e}_{u_x} \xleftarrow{\$} A$; $\beta_{u_x} \xleftarrow{\$} A$
$\vec{e} \leftarrow \vec{e}_1 \circ \vec{e}_2 \circ \cdots \circ \vec{e}_n$
$\beta \leftarrow \beta_1 \circ \beta_2 \circ \cdots \circ \beta_n$
return $(\vec{e}, \beta)$

$\underline{\textbf{proc UO}}$
$\mathcal{A}^{\text{input}(u_x, \alpha_x, p_x)}$
$\vec{e}_x, \beta_x \leftarrow \mathcal{A}^{\text{view}^{\vec{\pi}_q}_{\textbf{refut.}}()}$
return $\mathcal{A}^{\text{validate}_{UO}(g)}$

Figure 28: Game G_{UO} (Unobservability).

unobservability, $\mathcal{A}$ must not be able to distinguish if $\vec{e}_x \leftarrow \pi_{q,1}(\vec{\alpha}_x), \beta_x \leftarrow \pi_{q,2}(\vec{\alpha}_x)$ given $\alpha_x \in \vec{\alpha}_x, f^\star(\alpha_x) = u_x$, or, given $\alpha_x \notin \vec{\alpha}_x, f^\star(\alpha_x) \neq u_x, e_x \xleftarrow{\$} A$.

In addition to the general observation, we assume that $\mathcal{A}$ is able to learn U_f, Q_f and P_f. This allows $\mathcal{A}$ to learn the user identifiers associated with the outcome values, partitions of the overall outcome set $\vec{e}_x$ and information about the number of elements each user has contributed in the outcome set, which might or might not help to find said relation.

Remark 2. *The previous definitions of the privacy games is for individual privacy, where we assume u_x to be fixed. For group privacy, we assume $u_x \in \vec{u}_g$, where $\vec{u}_g$ is a group of fixed user identifiers. Then, each objective may be extended to find elements $e_x \in \vec{e}_x$ that relate to any u_x pertaining to the group of $\vec{u}_g$. For example, if $\mathcal{A}$ was about to infer if any e_x is related with e_y, where $f(e_x) = u_x$, $f(e_y) = u_y$ and $u_x, u_y \in \vec{u}_g$, then this would violate group privacy.*

3.8.3 Relations between Privacy Notions and Privacy Games

In this section, we further address arguments of equivalence/non-equivalence between any previously defined privacy notions (*i.e.* the notions presented in [72] and the notions presented in [358]) and our privacy games in addition to Section 3.7.7. The

previously presented privacy games are equivalent with our defined privacy notions and aim to closely follow the privacy notions of Bohli and Pashalidis [72]. Yet, in this context, we do not aim to show formal equivalence between our notions (and their representations as games) and the notions of Bohli and Pashalidis [72]. Our intention is to ensure equivalence of the underlying concepts and meaning of the privacy notions and the privacy games (*e.g.* game G_{SU} should 'objectively' represent the notion of *strong unlinkability*), but we do not require a formal (*i.e.* mathematically provable) one-to-one equivalence. (We identify such formal proofs as potential future work.)

Overall, the representation of the privacy notions in [358] and [72] is textual — definitions are stated informally without strict formal bindings. In contrast, both presentations assume a clear formal system model (where our system model follows closely those of [72]). To evaluate the 'outcome' of a privacy notion, [358] uses the term 'distinguish', while [72] uses the term 'leakage'. These terms are further used in the definitions of the privacy notions as informal arguments. However, both presentations do not formally define 'measures' for these terms which can be used to mathematically evaluate the outcome of a privacy notion. As a consequence, the loose bindings of the privacy notions with respect to the system model allow a 'liberal' interpretation of these notions. Thus, any potentially divergence of the interpretation of the notions (and, thereof, definition of our privacy games) are due to understanding of these definitions by the author.

In the literature a wide variety of privacy notions is present, with many of these privacy notions being 'informally equivalent'. This means, that while these notions may not have a proof of equivalence, they represent the same underlying meaning. For example, there may be multiple ways of formally defining the notions of unlinkability. However, in contrast, the contextual understanding of the word 'unlinkability' is, generally, the same in these cases.

With our games we want to represent an 'accumulation' of commonly accepted privacy notions and privacy properties. As such, our privacy games aim to represent a 'minimal' subset of operations that are required to achieve the outcomes of a privacy notion. A strict 'formal' one-to-one matching is in this context not necessary. However, a 'meaningful matching' is desired. The representation of our privacy notions in the form of games is motivated by our aim to bind the contributions of [358] and [72] into a common framework. While the contributions of Pfitzmann and Köhntopp [358] is motivated to define clear terminology and terms, Bohli and Pashalidis's [72] contributions are to presents their privacy notions in a formal setting and to show relations between those. The understanding of the privacy notions is yet not simply and clear. As a consequence, our goal is to bridge that understanding for researchers of such notions (and, more widely, researchers within the field of privacy). In this context, our privacy games offer an alternative to the purely textual representation of other presentations. Overall, our aim is to foster engagement and contributions within this context to generate new insights and enable better analyses of privacy-preserving applications.

3.9 Summary

The definition and representation of privacy and notions that reason about properties of privacy requires care. That is because privacy, by its very nature, is multi-dimensional and multi-faceted, and can mean different things to different people. Consequently, in recent years, various notions for privacy have been proposed. To provide a clear understanding of privacy (*e.g.* via privacy notions), its properties and implications to systems, it is important to accurately model adversarial actions. Several approaches have emerged, whereas we focus on the game-based approach.

In this chapter, we set out to provide a clear and concise understanding of notions for privacy-preserving data release. Precisely, we build upon the syntactic notions of other authors and formulate those notions into privacy games, following the game-playing technique of provable security. We extend the previously introduced framework of Bohli and Pashalidis [72] by the notion of unobservability — a notion that captures the capability of an adversary to distinguish if a system invocation has taken place or not.

With the definition of these games, we aim to provide an alternative viewpoint to the purely textual representation of other authors. This, in turn, should help to clarify the understanding of, and relationships between, different privacy notions; it may foster analyses of privacy-preserving applications and consequently help to generate new insights. In addition, the privacy games give an unambiguous understanding of adversarial actions (in order to win a game and, therefore, break a notion). In the definition of the privacy notions and games, we take into consideration the different requirements of privacy protections for individuals and groups. Overall, many currently existing models for privacy focus on the preservation of privacy in centralised environments. Various use cases, however, have shown the importance for concise privacy models and notions in distributed environments. Our notions, games and system model are therefore designed for distributed environments (yet, are still applicable to centralised systems).

Overall, the contributions of this chapter aim to encourage discussion and analyses of syntactic privacy notions and our abstraction via games may be used to create novel tools to automatically analyse privacy properties of privacy-preserving applications. In this context, the proposed games should be seen as an *alternative* approach to foster the understanding for experts such as privacy researchers and academics. The game-based approach has already been widely used in the cryptography community to provide simpler and more easily verifiable proofs. Likewise, our motivation is simplify understanding and applicability of the privacy notions to reason about the associated privacy properties. Automatic tools may further engage non-experts such as software developers and system designers to analyse their applications and generate new insights of the relevant notions of privacy.

In the next chapter, we evaluate and analyse the privacy games presented in this chapter. We discuss general limitations of the approach, state generic probabilities to win a game (*i.e.* break a notion), and present relationships between the privacy games. We present a first approach of an automated tool for our privacy games. Predominantly for non-experts, we state policies for the selection of a 'fitting' privacy game.

Further, we present the application of our privacy games to a case study on content-based clustering recommender systems and collaborative-filtering based classification recommender systems.

» *We think in terms of a 'privacy budget'. After a series of queries exhausts the privacy budget, that's it! You have to kill the user.* «

— Bill Howe in *Communicating Data Science Results*

4 EVALUATION AND ANALYSIS OF SYNTACTIC PRIVACY GAMES

Contents

Publication Data

This chapter is based on the following publications:

(1) Robin Ankele and Andrew Simpson. Analysing and Evaluating Syntactic Privacy Games via a Recommender Systems Case Study. In: Proceedings of ATC 2019. 2019. [36]

(2) Robin Ankele and Andrew Simpson. Extended Abstract — Analysis and Evaluation of Syntactic Privacy Notions and Games. In: Proceedings of PST 2018. 2018. [32]

In this chapter, we present an evaluation and analysis of the privacy notions and privacy games defined in the previous chapter. Our investigation starts by modelling two classes of adversaries according to which we analyse the privacy games. Next, we present some limitations of our underlying system model and adversarial model as well as general limitations of our approach. Following from this, we present probabilistic bounds of the likelihood of an adversary winning a game (*i.e.* breaking an associated notion). Next, we present a first implementation of a possible tool to automate privacy-preserving analyses via privacy games. Then, predominately for non-experts, we state policies for the selection of a 'fitting' privacy game. Our investigation is motivated through a case study based on recommender systems. In this context, we show how the game-based definitions have the potential to interconnect privacy implications and to reason about privacy properties of content-based clustering recommender systems and collaborative-filtering based classification recommender systems. Finally, we summarise the contributions of this chapter.

4.1 Overview

The sheer number of privacy notions proposed by academics in recent years (as discussed in the previous chapter and in Part I and II of Chapter 2) are not helpful for non-experts such as software developers and software architects. Also, young academics and privacy researchers new to the field of privacy (and the swarm of privacy notions) may struggle to: (a) get an understanding of the privacy notions and (b) select an appropriate notion for their system model. If anything, the sheer number of notions contributes to the complexity of selecting an appropriate way to reason and argue about privacy properties in academic and real-world systems.

For non-experts, it is highly unlikely that they are reading through a vast amount of academic literature to find an appropriate way to describe the privacy properties of their systems. Thus, privacy considerations are often ignored or wrongly communicated. Software developers, whose aim is to leverage software systems in order to automate, solve and optimise their use cases, are often not able to find meaningful ways to compare different systems.

For experts, the number of privacy notions may hinder analysis of privacy-preserving systems. The lack of a common, standardised framework prevents comparison between notions. Further, it prevents the generation of much needed insights and relationships between privacy notions.

In the context of privacy systems, predominantly, there is a missing framework of well analysed privacy notions and their relationships to each other. So far, to support both groups (non-experts and experts) different approaches have been proposed to: (a) enhance a better understanding of privacy; (b) educate non-experts; (c) regulate privacy considerations between parties; (d) adapt considerations of privacy early in the design process of an application; and (e) maintain privacy throughout the whole lifecycle of an application. Examples addressing these points include (partly discussed in Chapter 2):

(a) **Privacy Standards**: such as ISO/IEC 20889 [9], ISO/IEC 29100 [2, 8] and ISO/IEC 29101 [7].

(b) **Regulations**: such as the GDPR [177] and HIPAA [1]

(c) **Privacy Principles and Guidelines**: by the IAPP[1] [101], the OECD[2] [336], and the APEC[3] [37]

(d) **Privacy by Design**: comprises 7 core principles to be taken into account throughout the whole engineering process which was initially developed by Ann Cavoukian [100].

Moreover, mainly on the academic side, the probabilistic notion of differential privacy [159] has emerged as a popular measure for systems that support interactive data release. Consequently, there exists a portfolio of mechanisms to achieve differential privacy. A complementary approach to the probabilistic notions was introduced

[1] International Association of Privacy Professionals (IAPP) — https://iapp.org/
[2] Organisation for Economic Co-operation and Development (OECD) — http://www.oecd.org/
[3] Asia-Pacific Economic Cooperation (APEC) — https://www.apec.org

in the previous chapter with the focus to represent syntactic privacy notions such as anonymity, unlinkability, pseudonymity and unobservablility in the form of games.

The games represent an alternative viewpoint to the purely textual representation of other authors. Games have been used widely in the cryptographic community to simplify and enhance the readability of proofs. Further, the game-based techniques have generated new insights and fostered analysis between cryptographic primitives.

For non-experts, these games can be incomprehensible without any practical context. As a consequence, this chapter is dedicated to provide means to combine, enhance and extend these theoretical contributions by showing that the game-based definitions have the potential to interconnect privacy implications and to reason about privacy properties. We motivate our analysis through a case study based on recommender systems.

For experts, this chapter presents an analysis and limitations of the privacy games. For both groups, we present a first approach to enable automated privacy-preserving analyses via our privacy games software tools. Taken together, these software tools should help to foster analysis between privacy notions and generate new insights with respect to the usability, flexibility and applicability of privacy notions and games in privacy-preserving applications.

4.2 Analysis of Privacy Notions and Privacy Games

In the following analysis, we elaborate on the capabilities of adversaries and generic limitations of the games. Then, we assess some generic success probabilities associated with an adversary playing the privacy games.

4.2.1 Adversary Classes

In general, any adversary $\mathcal{A}$ may be limited in her ability to infer knowledge of a system. As such, she may be able to observe *any* interaction of a user with the system (*i.e.* input and release of data), *only* the released outcome, or certain aspects and properties of the systems release. (The privacy notions of Bohli and Pashalidis [72] and our games have been modelled to abstract this within the release (*i.e.* by regulating access to sets U_f, Q_f and P_f via interfaces).)

We consider $\mathcal{A}$ to be of one of two classes[4]: either of class $\mathcal{A}_{release}$ or of class $\mathcal{A}_{in/out}$. The former, a release adversary, is only able to access the released results of the system $\vec{e}, \beta$. The latter, an in/out adversary, is able to input data α_x into the system, observe other inputs $\vec{\alpha}$ to the system, and observe the released results of the system $\vec{e}, \beta$. Figures 29 and 30 visualise these kind of adversarial classes. We omit adversaries of the type, $\mathcal{A}_\phi$ (system adversaries), thus treating a system, ϕ (and its internal operations), as an assumable secure and privacy-preserving black box model. Such a kind of secure system may be achieved by trusted and hardware-enforced secure entities such

[4]We recognise that there may be other classes of adversaries. To simplify our analysis, we consider these two classes, which, in general, should cover a wide range of adversary types.

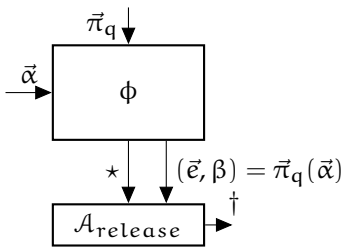


Figure 29: An adversarial model of type: release adversary[5]. This kind of adversary only obtains access to the system release $\vec{e}, \beta$.

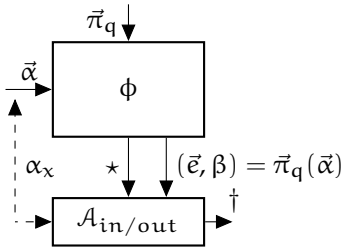


Figure 30: An adversarial model of type: in/out adversary[5]. This kind of adversary obtains access to the inputs $\vec{\alpha}$ as well as the system release $\vec{e}, \beta$.

as a SGX-based TRE (see Chapter 6) or any other strong cryptographically-protected system.

4.2.2 Generic Limitations of the Privacy Games

We recognise that our approach (*i.e.* our adversary, system and privacy model; privacy notions; and privacy games) may not be universally applicable to cover all privacy aspects of (privacy-preserving) systems. As such, we identify some generic limitations, which include:

1. **Release of multiple elements**: While our system model, in general, supports the release of only the aggregated value β, the privacy notions and our games are solely defined over the vector $\vec{e}$ (*i.e.* β is not included the definition of the privacy notions / games).

2. **Released elements $e_i \in \vec{e}$ uniquely related to u_i**: In order to apply our privacy notions and games directly, each released element $e_i \in \vec{e}$ must be uniquely mapped to exactly one user identifier, u_i. There may be multiple elements in $\vec{e}$ which map to the same u_i, but there must not be an element $e_i \in \vec{e}$ which maps to two different user identifiers u_i, u_j, where $u_i \neq u_j$.

3. **Privacy level dependent on type of (privacy-preserving) transformation function $\pi_{q,i}$**: Our privacy games are defined to outline the necessary steps for an adversary to break an associated notion. However, the level of privacy achieved is dependent on the type of the transformation function $\pi_{q,i}$ (*e.g.* $\pi_{q,i}$ may be a secret permutation or perturbation-based).

[5] $\star$ indicates the knowledge $\mathcal{A}_{release}$ or $\mathcal{A}_{in/out}$ is given access to; further, $\dagger$ represents the learning goal of $\mathcal{A}_{release}$ or $\mathcal{A}_{in/out}$. These values change depending on the desired notion system ϕ is considered to achieve. A mapping of potential values associated to each privacy notion and game is given in the last two domains of Table 4.

4. **Privacy model may not cover all privacy aspects**: While the privacy notions and games abstract a wide range of privacy issues, they may not cover all aspects of privacy (*e.g.* query privacy in the sense of differential privacy[6]).

5. **Adversary types and assumptions**: We consider an adversary who aims to learn the mapping function f to identify user identifiers from release elements e_i. There may be other privacy implications: for example, the elements e_is may be by itself sensitive to release. Moreover, in this chapter we do not consider system adversaries, which may learn information about sensitive attributes from, for example, side channels[7]. Also, in this chapter, we assume that adversaries are semi-honest and follow the rules of the games. In real world scenarios, it is not possible to always make such an assumption.

6. **Rigorous mathematical treatment and proofs**: While our system model, privacy notions and games follow a rigorous mathematical framework (to abstract a wide range of real world systems), we do not formally prove systems secure (with respect to our games). This is partly due to missing definitions of 'idealistic' instances (of privacy-preserving systems) with regards to the privacy properties that are abstracted by our games (*e.g.* an idealistic instance with regards to unlinkability would always return unique 'unlinked' outcome values). Moreover, for example in our case study, we do not consider a concrete implementation of our privacy-preserving transformation function $\pi_{q,i}$ (*e.g.* $\pi_{q,i}$ is a perturbation function which adds random noise according to a given distribution). This is to remain abstract from concrete implementations to cover a wide range of applications (by abstracting the concrete mechanism and considering that data may be transformed; and, as such, be privacy-preserving).

4.2.3 Generic Success Probabilities associated with the Privacy Games

In order to provide release privacy, a system ϕ must deploy a privacy-preserving transformation function $\vec{\pi}_q$. In our further analysis, we assume that $\vec{\pi}_q$ is a privacy-preserving identity mapping. As such, it does not alter the input values $\vec{\alpha}$, but obfuscates the user identifier, u_i, from the released elements, $\vec{e}$. However, $\vec{\pi}_q$ may be a set of any kind of functions: for example, transforming multiple input values $\vec{\alpha}$ to a single outcome value, e_i. Thus, the probability of an adversary of kind $\mathcal{A}_{release}$ or $\mathcal{A}_{in/out}$ to break any of our privacy notions needs to be assessed with this kind of transformation function $\vec{\pi}_q$ in mind.

We assume that $|\vec{e}| \geqslant |\vec{u}|$, with $|\vec{e}| = m$ and $|\vec{u}| = n$. Both m and n are 'large' integers — of an order such that an adversary who has access to modern computational hardware should not be able to gain any advantage compared to randomly guessing[8].

[6] Differential privacy ensures that the addition of a element to a database (*i.e.* participation of an individual) doesn't change the query result significantly. Chapter 2 elaborates in more detail on this definition. In the context of our syntactic privacy games, a combination of the notions PH (participation hiding) and UO (unobservability) may closely resemble such privacy requirements.

[7] Information gained from a side channel (attack) is by targeting a concrete implementation rather than any weakness in the algorithm.

[8] If the release set $\vec{e}$ and the set of user identifiers $\vec{u}$ are sparsely populated an adversary may simply try any combination of variants to win a game. In this context, the elements pertaining to the sets $\vec{e}$, $\vec{u}$ should be

Then, for a release adversary $\mathcal{A}_{release}$, under the assumption of a fixed user identifier u_x, the following probabilities can be observed:

(a) $Pr[\mathcal{A}^{\star}_{release} \Rightarrow \text{true}] = \frac{1}{|\vec{e}|-q}$, for games abstracting the notions $\star \in \{SA, PH\}$ with learning goal $f(e_x) = u_x$.

(b) $Pr[\mathcal{A}^{\star}_{release} \Rightarrow \text{true}] = \frac{1}{|\vec{e}|-q} * \frac{1}{|\vec{e}|-q}$, for games abstracting the notions $\star \in \{SU, WU\}$ with learning goal $f(e_x) = f(e_y) = u_x$ (assuming independence in the selection of e_x and e_y).

(c) $Pr[\mathcal{A}^{\star}_{release} \Rightarrow \text{true}] = \frac{|\vec{e}_{u_x}|}{|\vec{e}|-q}$, for games abstracting the notions $\star \in \{PS, AN, WA\}$ with learning goal $f(\vec{e}_{u_x}) = u_x$.

(For all games, q indicates the number of queries that $\mathcal{A}_{release}$ has submitted to the system ϕ. $|\vec{e}|$ denotes the size of the release elements — capturing the number of elements ϕ releases. Similarly, $|\vec{e_{u_x}}|$ accounts for the number of elements included in the release for a user u_x.)

For game G_{UO} (unobservability), we would need to deploy a different measure to assess the success probability to win said game. Concretely, such a measure could be the distinguishing advantage — though this requires the definition of an 'idealised' instance of the privacy-preserving system with regards to the aforementioned game. To assess the distinguishing advantage, it is evaluated if a 'real' instance is able to satisfy these guarantees (those outlined by the 'idealised' instance). To give an idea of such an approach, we roughly estimate $\mathcal{A}$'s probability to win game G_{UO}: we assume a release adversary ($\mathcal{A}$ is only give access to the outcome $(\vec{e}, \beta)$) and assume an 'idealistic' instance, where the target element $(\vec{e}_{u_x}, \beta_{u_x})$ is replaced by a random element from the same parameter space; the success probability is then estimated to be $Pr[\mathcal{A}^{\star}_{release} \Rightarrow \text{true}] = 0$, with $\star \in \{UO\}$. For a 'real' instance we assume that the generation of a replacement value would leak information (or may be detectable).

In the case of an in/out adversary, $\mathcal{A}_{in/out}$, the transformation functions $\vec{\pi}_q$ need to be strictly more privacy-preserving, since $\mathcal{A}_{in/out}$ has additional access to the inputs $\vec{\alpha}$ of a system, ϕ. Depending on the data within the system, $\mathcal{A}_{in/out}$ might submit any outlier elements α that are easily recognisable in the case of a privacy-preserving identity mapping. In such cases, $\vec{\pi}_q$ must transform the released elements (*e.g.* by a perturbation-based privacy mechanism). Otherwise, the same probabilities as in the case of a release adversary apply.

In some application scenarios and systems it is neither possible nor feasible to release elements with a large privacy margin. Therefore, to prevent an adversary from learning too much information, while still being able to maintain a smaller privacy margin and achieving certain privacy guarantees, it is possible to limit the access to any validate interfaces $\text{validate}_{\star}$. In this case, data must not be shared after a threshold of queries is reached in order to prevent loss of any individual's privacy.

selected from a range with cardinality comparable to modern cipher parameters (*e.g.* 2^{112} [50]). Under the assumption that the adversary does not know any or only limited information about the contents of $\vec{e}$, $\vec{u}$, he/she should not gain any advantage compared to just randomly picking any element from the range.

4.3 Software Tools

In recent years, many software tools have fostered automation and generated valuable new insights in various software systems. To bridge the gap between the theoretical contributions of Chapter 3 and our more practical advances in the following chapters, we developed a software implementation of the previously introduced privacy games. These software tools should foster the understanding of privacy properties for non-experts (by engaging developers with a 'familiar' environment). Further, these tools may be used by experts to develop advanced software tools to automate the analysis of privacy-preserving applications. In this context, these tools should help to clarify relations between privacy notions and, in the long-term, help to engage the IT community to gain a better understanding of privacy.

Our software tool library is mainly targeted at the following demographics:

- **Privacy experts**: who are looking to develop software tools to automate the analysis of privacy-preserving applications and to generate new insights and relationships between privacy notions.
- **Software developers and system designers**: who are looking to get hands-on tools to educated and familiarise themselves with privacy tools given a common playground.

For privacy experts, our tools may help with the analysis of privacy notions to extract the underlying principles and insights. Software developers and system designers shall be able to use the insights to create secure and privacy-preserving applications. As such, it is on the research community to simplify the understanding and foster analysis and comparisons between privacy notions such that these can be better understood and used in practice.

Our contributions, *i.e.* the insights generated in Section 3.7, the privacy games presented in Section 3.8, the software tools of this section, the policies for the selection of a privacy game (presented in the next section) and the case studies presented in Section 4.5 are the first steps to combine the contributions of [358] and [72] to foster engagement and simplify analyses. Our software tools:

- should be seen as a bridge between: (a) the terminology of Pfitzmann and Köhntop [358] and (b) the formal treatment and analysis (*i.e.* relationships) of Bohli and Pashalidis [72];
- provide a playground to experiment with the abstract privacy games;
- should help the community to provide analyses of our privacy notions and those of other authors via theorem provers and potential other methods.

In this context, to allow others to experiment with the privacy games and to extend our contributions, our privacy games library is available under open-source license at https://github.com/robinankele/privacy-games. In addition, in Appendix D we append the source files of our software library. The stand-alone library, is written in python 3.7 and consists of an implementation of each privacy game and a small range of so-called privacy primitives. The privacy primitives can be used to plug into the privacy games as transformation functions $\vec{\pi}_q$.

4.4 Policies for the Selection of a Privacy Notion and Privacy Game

In this section, we clarify how non-experts can select fitting privacy notions and games to analyse their systems. We note that these 'policies' reflect the opinion of the author and have not been validated in any meaningful way. Nevertheless, the selection of a 'fitting' privacy notion / game is often inherently given from the application context, as described below. As such, these policies should be seen as an informative addendum and future work may validate our conjectures.

4.4.1 Generic Policies

In order to address the question of which 'privacy notion/ privacy game' a non-expert should choose in order to reason about their privacy system, it is necessary to establish a baseline which can be done by addressing the following questions:

1. Clearly define / select a system model specifying all entities, objects, subject and relations.
2. Clearly define / select the adversarial capabilities (*e.g.* via approaches outlined in Section 3.1 and our approach defined in this dissertation).
3. Define / select privacy notions (*e.g.* those defined by [72], [358] and those derived in this dissertation).
4. Select a privacy primitive[9].
5. Prove secure via concise logical reasoning or mathematical arguments.

The contributions of this chapter, as well as those in other parts of this dissertation, may help non-experts to select a fitting notion. By accumulating the required information (via the above questions) the choice to select a specific privacy notions should be limited. Further, the following approaches may be used.

4.4.2 Selecting a Privacy Notion / Privacy Game

Due to their structure, this selection is inherently given from the definitions of the privacy notions. In Table 4 the underlying privacy notions of our privacy games are summarised and assigned to three 'filtering' conditions:

1. (Behaviour) Category
2. Learning Goals
3. Knowledge of an Adversary

[9] A privacy primitive, in this context, is a privacy mechanism that can be applied to transform sensitive data into non-sensitive data. Examples are defined in Section 2.9.2.

On the one hand, it allows our privacy games to be classified into four categories: (a) single elements, (b) multiple elements, (c) element groups and (d) behavioural. As such, depending on the context of the privacy release of the system under question, either of the notions presented in the groups applies. (We note that this categorisation is not exclusive, since privacy notions pertaining to, for example, the single elements category, may also be used to analyse single elements within a group release.)

On the other hand, the privacy games can be selected according to the learning goals of a potential adversary. For each of the previously described categories, a different learning goal of the adversary applies. In this context, by selecting a privacy notion within a category, different levels of privacy guarantees (within the group) can be explored. This means, for example, in the group 'multiple elements' the learning goal of an adversary could be to distinguish if any two elements, e_x, e_y pertain to the same user identifier, *i.e.* $f(e_x) = f(e_y)$. Yet, in this category two notions *strong* and *weak* unlinkability are available, whereas a system designer can choose either — for example, via our last category: the knowledge of an adversary.

In contrast to the above selection criteria, we briefly discuss two other approaches based on: (a) any knowledge a system leaks (voluntarily) and (b) any (potential) a priori knowledge an adversary has. As indicated in Table 4, each privacy game has limited access to a certain set of system properties such as the participant set, the user frequency set and any linking relations. While for case (b) we assume that an adversary, by whatever means, is able to attain access the knowledge conveyed by previously mentioned system properties, for case (a) we assume that the system provides access to these interfaces voluntarily — it covers the case that any a priori knowledge of an adversary can not be easily assessed, thus, for staging / testing environments, system designers can use these provisions to analyse their systems. In this context, a system that achieves a privacy notion does not leak any information beyond what the adversary is given access to anyway.

This means that a system designer can choose the privacy notions according to the information that 'leaks' from the system. In other words, the privacy notions can be chosen according to the context of what kind of infomation is appropriate to be revealed for a given system. For example, if a system should achieve 'unlinkability', it is acceptable if the participants, U_f, are known (*i.e.* strong unlinkability) and, in addition, the amount of elements per participant, Q_f, within a release are known (*i.e.* weak unlinkability).

4.5 A Case Study: Recommender Systems

After having presented an abstraction of privacy notions via games, we are interested in showing their applicability. As such, in the following, we utilise the privacy games and evaluate them in a case study based on recommender systems. Moreover, we analyse our approach: concretely, we present limitations and evaluate probabilities to successfully break said games.

4.5.1 Preliminaries of Recommender Systems

With the rapid explosion of new developments and technologies for web-based services in recent years, there has been, consequently, an increase in the complexity and choice a user faces when accessing these services. This significant growth of services has the potential to lead users to struggle to find the information they are looking for and can make the selection of a product difficult and time-consuming. The deployment of a recommender system has proven to be an efficient method to tackle this problem. The suggestion of similar products may allow a user to make a choice from a wide product range. Hence, it can reduce the information overload a user has to deal with while searching for similar items. Such systems also help to cross-sell additional products that a user might not have thought of, but may be interested in buying.

While several kinds of recommendation methods [249] have been proposed, within this case study we focus on the two common approaches of: *(a) content-based recommendations* and *(b) collaborative recommendations*. The former method generates recommendations based on similar items; the latter generates recommendations based on user collaborations or similar user behaviour[10].

The 'efficiency' of a recommendation system is highly dependent on a couple of factors (and, as such, the choice to use and potentially re-use such a system): accuracy of recommendations, diversity and serendipity of recommended items, persistence of the recommendation, privacy of individuals and group of individuals, user demographics, and robustness and trust in the recommendation process. In order to make recommendations that are both accurate and diverse, it is necessary to collect a substantial amount of user data, including their purchase history and their preferences. Consequently, such collections raise concerns about an individual's privacy (or the privacy of a group / cluster of users). Examples include: vulnerabilities in collaborative filtering based recommendation algorithms (*e.g.* the k-nearest neighbour algorithm) shown by Calandrino *et al.* [90]; privacy concerns for users that rate elements across disjoint domains shown by Ramakrishnan *et al.* [368]; and privacy concerns through user control shown by Zhang *et al.* [451].

4.5.2 Content-based Clustering Recommender Systems

We consider a system that recommends products based on previous transactions. As such, the system takes as input user transactions and releases clusters of similar transactions. Using these clusters, we can recommend products to other users pertaining to the same cluster.

Examples of clusters in this context include (but are not limited to): transactions below a certain price threshold, transactions within a certain store, and transactions that involve a certain category of products. Figure 31 illustrates an example cluster.

To clearly represent and illustrate our recommendation method, we define some properties pertaining to transactions. Thus, each transaction has relationships to certain properties in the form of a function. The properties must include a *unique* user

[10] We know, for example, that users in a specific class or cluster will have similar interests. Thus, it is natural to recommend products bought by users in a cluster to others in the same cluster.

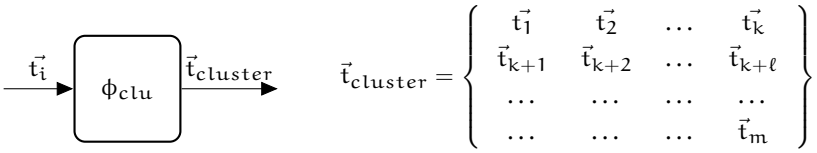


Figure 31: A recommender system deploying content-based clustering: it takes transactions $\vec{t_i}$ as input and releases a cluster $\vec{t}_{cluster}$ of similar transactions. It is the case that $\vec{t_i} \in \vec{T}$ and $\vec{t}_{cluster} \subseteq \vec{T}$, where $\vec{T}$ denotes the set of all transactions and $m = |\vec{t}_{cluster}|$. Each $\vec{t_i}$ must have a unique member $\vec{t_i}.user$ and $\vec{t_i}.product$.

identifier and a product identifier, and may also include a price, a rating or a store identifier.

Our recommendation method releases clusters as an outcome. Within a cluster, multiple transactions may pertain to the same user; however, each single transaction must pertain to exactly one user. For each product a user buys, a mapping associated with the transaction is stored in the recommender system. After a certain threshold of transactions, the system may be queried to release a cluster of transactions. Using these clusters, recommendations can be computed.

The privacy implications of such a recommendation method are numerous. For example, an analyst may be permitted to learn a cluster of transactions. In contrast, if we consider a potential malicious analyst (denoted by $\mathcal{A}$): $\mathcal{A}$ must not be able to learn the user who conducted the transaction; further, $\mathcal{A}$ must not be able to link transactions pertaining to the same user. We apply the privacy notions and games on the released clusters.

4.5.3 Collaborative-filtering based Classification Recommender Systems

We motivate our second collaborative-filtering (CF) based classification approach by the k-nearest neighbour (k-nn) attack. That is to say, with our approach, we aim to illustrate a CF-based classification algorithm that is able to withstand this kind of attack. As such, we consider a more complex, two-step method[11] involving classification as a second step after clusterisation. As such, we reuse the released clusters from the recommendation algorithm described previously.

As a first step, we generate the transaction clusters. In the second step, we use these clusters as 'classifiers' — in order to assign (*i.e.* classify) if a new user added to the system pertains to a certain class of k-nearest neighbours. As the last step, we use a simple majority function over the k-nearest neighbours to recommend products to the users.

[11] To understand our reasoning of this two-step recommendation, we remind the reader of the k-nn attack [90]: under the assumption that an adversary knows a certain subset of products, $P_{overlap}$, bought by a user, an adversary generates k fake users. All of these fake users are then assigned the same subset of products as the target user. Thus, when calculating the k-nearest neighbours for the targeted user all of these fake users are included (with high probability, due to the number of overlapping products). By querying for a recommendation all the products bought by the targeted user are revealed (with high probability, all other products except $P_{overlap}$ are products bought by the target user) — a severe break in the targeted user's privacy.

Figure 32 illustrates this multi-step recommendation method. As in the system described previously, our multi-step recommendation system remains filled with transactions. We release clusters as a first step and k-nearest neighbours as the second step. From the k-nearest neighbours, we release recommendations of single products. The privacy implications of this system are as follows: the clustering implications remain the same; for the classification step, we want to defy the k-nearest neighbour attack. We argue that with our method we do not generate the k-nearest neighbours from users who bought similar products, but via users that performed similar transactions — transactions within the same transaction cluster. As such, knowledge of products bought by a user does not reveal the k-nearest neighbours. Yet, we generate the k-nearest neighbours using the publicly released clusters. However, since we require that an adversary must not be able to learn the user identifier associated with a release transaction within any transaction cluster, revealing the clusters is fine. The privacy notions and games can be applied to the released k-nearest neighbours in order to reason about privacy.

4.6 Analysis of Recommender System via Privacy Games

4.6.1 Analysis of Content-based Clustering Recommender Systems

Consider a content-based clustering recommendation system, ϕ_{clu}. As inputs, $\vec{\alpha_i}$, to such a system, we consider transactions, $\vec{t_i}$. A transaction is a vector of multiple input values (*i.e.* a transaction has properties such as price and user identifier); however, we treat each single transaction as a single input value, α_i. In the next step, the recommendation system, ϕ_{clu}, performs a set of transformation functions, $\vec{\pi_q}$ — the clusterisation method. As a result of the clusterisation, ϕ_{clu} releases a vector of transactions: cluster $\vec{t}_{cluster}$, denoted by the output value vector, $\vec{e_i}$, for release query, q_i.

As discussed previously, we require that each transaction $\vec{t_i}$ maps to a unique user identifier u_i — in our case, the user who bought the product associated with the transaction $\vec{t_i}$. We represent the mapping of the properties of a transaction as functions of the form $f : \vec{t_i} \rightarrow u_i$. Importantly, we consider the mapping between the user identifier, u_i, and the transaction, $\vec{t_i}$, released in the output cluster, $\vec{t}_{cluster}$, as *sensitive*. Thus, we want to prevent any adversary $\mathcal{A}$ from learning this mapping or any other linking relation of such a mapping to other transactions. There may exist other mapping functions such as the mapping of the price to a transaction that remains *non-sensitive* and may be revealed to an adversary or analyst to perform recommendations.

According to our system model, we consider β to be a released recommendation, which is calculated using the results $\vec{t}_{cluster}$ of the clustering step $\pi_{q,1}$ via a recommendation function, $\pi_{q,2}$. Together, $\pi_{q,1}$ and $\pi_{q,2}$ form the vector of transformation function $\vec{\pi_q}$ of ϕ_{clu}. We require that both $\pi_{q,1}$ and $\pi_{q,2}$ are privacy-preserving (as such, not to reveal any of the previously as sensitive classified elements or mappings).

Table 7: Analysis of a content-based recommender system ϕ_{clu}.

Notion	✓/−	Comments
SA	✓	$\mathcal{A}$ only observes $\vec{t}_{cluster}$ and is given no additional information. By the requirement that f is not accessible, $\mathcal{A}$ does not gain any advantage to choose tuple (t_x, u_x) for interface $validate_{SA}$.
PH	✓	$\mathcal{A}$ observes $\vec{t}_{cluster}$ and is given access to $\lvert U_f \rvert$. $\mathcal{A}$ does not gain any advantage to select tuple (t_x, u_x) for interface $validate_{PH}$.
SU	✓	$\mathcal{A}$ observes $\vec{t}_{cluster}$ and is given access to U_f. $\mathcal{A}$ does not gain any advantage to select triple (t_x, t_y, u_x) for interface $validate_{SU}$.
WU	✓	$\mathcal{A}$ observes $\vec{t}_{cluster}$ and is given access to U_f and Q_f. $\mathcal{A}$ does not gain any advantage to select triple (t_x, t_y, u_x) for interface $validate_{WU}$.
PS	✓	$\mathcal{A}$ observes $\vec{t}_{cluster}$ and is given access to P_f. $\mathcal{A}$ does not gain any advantage to select tuple $(\vec{t}_{u_x}, u_x)$ for interface $validate_{PS}$.
AN	✓	$\mathcal{A}$ observes $\vec{t}_{cluster}$ and is given access to U_f and P_f. $\mathcal{A}$ does not gain any advantage to select tuple $(\vec{t}_{u_x}, u_x)$ for interface $validate_{AN}$.
WA	✓	$\mathcal{A}$ observes $\vec{t}_{cluster}$ and is given access to U_f, Q_f and P_f. $\mathcal{A}$ does not gain any advantage to select tuple $(\vec{t}_{u_x}, u_x)$ for interface $validate_{WA}$.

Over the released clusters we can define the sets U_f, Q_f and P_f as follows. The participant set, U_f, contains all user identifiers, u_i, of all released transactions, $\vec{t_i} \in \vec{t}_{cluster}$. The usage frequency set, Q_f, contains a mapping between the user identifier, u_i, and the number of transactions, $\#u_i$, pertaining to that user, of all released transactions $\vec{t_i} \in \vec{t}_{cluster}$. Finally, the linking relation set, P_f, partitions all released transactions $\vec{t_i} \in \vec{t}_{cluster}$ regarding their associated user identifiers, u_i.

The aim of a potential malicious analyst is to find or associate a user identifier to the transactions released in cluster $\vec{t}_{cluster}$. Considering the data to be released, Table 7 presents details of the privacy notions that can be achieved by playing our privacy games.

4.6.2 Analysis of Collaborative-filtering based Classification Recommender Systems

Consider a collaborative-filtering based recommendation system, ϕ_{col}. As discussed previously, ϕ_{col} is based upon the content-based clustering system, ϕ_{clu}, and takes as input a cluster of transactions, $\vec{t}_{cluster}$. We treat each single transaction $\vec{t_i} \in \vec{t}_{cluster}$ as a single input $\vec{\alpha_i}$. As the next step, the transformation functions $\vec{\pi}_q$ of ϕ_{col} execute the classification algorithm as described previously and release vectors of products, denoted by $\vec{e_i}$, for release query q_i and user u_i. Each row in this release pertains to one of the k-nearest neighbours of user u_i. Similar to the reasoning for ϕ_{clu}, the release products p_i still remain associated to their user identifier u_i via a function

$$f : \mathfrak{P} \mapsto \mathfrak{U}; p_i \mapsto f(p_i) \tag{9}$$

with $u_i = f(p_i)$, $\mathfrak{P}$ the parameter space of all products, and $\mathfrak{U}$ the parameter space of all user identifiers. Yet, ϕ_{col} does not release any user identifier u_i in plain, but as a pseudonym, n_i. Pseudonym n_i may pertain to any user identifier u_i within the set of all user identifiers, $u_i \in U_f$. In detail, the n_is map to the k-nearest neighbour of a targeted user, u_j. We require that any potential malicious analyst $\mathcal{A}$ must not learn

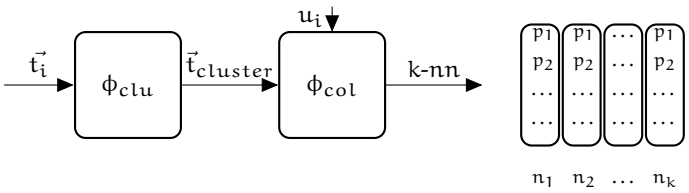


Figure 32: A recommender system deploying collaborative-filtering based classification: in the first step transactions $\vec{t}_i$ are clusterised into clusters $\vec{t}_{cluster}$; in the second step, for a user u_i, the products p_i of all k-nearest neighbours $(n_1, \ldots, n_k)$ are released. Recommendations can be generated via a majority vote over similar products p_i among all neighbouring users.

the mapping of function f (see Equation 9) and the pseudonym mapping described by a function

$$g : \mathfrak{U} \mapsto \mathfrak{U}; n_i \mapsto g(n_i) \tag{10}$$

with $u_i = g(n_i)$ and $\mathfrak{U}$ the parameter space of both user identifiers and pseudonyms. Concretely, f (Def. 9) is the mapping function of outcome elements to user identifiers of our system model and g (Def. 10) is a mapping function of pseudonyms n_i to the user identifiers u_i (where n_is are drawn from the same parameter space as u_is). We note that system ϕ_{col} only releases products p_is as outcome elements and that the associated pseudonyms n_i are presented here to provide a simplified understanding, due to the structure of the released products (*i.e.* p_is are ordered and reveal the linking relation P_f).

Again, according to our system model, we consider β to be a released recommendation. β is calculated using the resulting vector of products $\vec{e}$ of the k-nearest neighbours of a user u_i of the classification step $\pi_{q,1}$ via a recommendation function $\pi_{q,2}$. $\pi_{q,1}$ and $\pi_{q,2}$ form the vector of transformation function $\vec{\pi}_q$ of ϕ_{col}. We require that both $\pi_{q,1}$ and $\pi_{q,2}$ are privacy-preserving (so as to not reveal any of the previously as sensitive classified elements or mappings).

Over the released products of the k-nearest neighbours of user u_i, we define the sets U_f, Q_f and P_f as follows. All user identifiers, u_i, of the released products are included in the participant set, U_f. Similarly, the usage frequency set Q_f contains a mapping $(u_i, \#u_i)$ of all user identifiers, u_i, and the number of products, denoted by $\#u_i$[12], bought by that user (and listed in the released products). Finally, the linking relation set P_f partitions all released products p_i regarding their associated user identifiers u_i[13].

Again, the aim of a potential malicious analyst is to find or associate a user identifier to the products released of system ϕ_{col}. Considering the data to be released or inferable, Table 8 presents details of the privacy notions that can be achieved by playing our privacy games.

[12] Even though we denote the number of products, $\#u_i$, in the same style we use to denote single elements, by $\#u_i$ we mean the cardinality of products bought by user u_i.

[13] In fact, due to the release mode of ϕ_{col}, P_f may always be identified, even if it is not accessible through interface $\mathfrak{I}_{P_f}$, by simply observing the columns that ϕ_{col} releases.

Table 8: Analysis of a collaborative-filtering based recommender system ϕ_{col}.

Notion	✓/−	Comments
SA	✓	$\mathcal{A}$ observes $\vec{e_i}$ and can infer P_f. Yet, $\mathcal{A}$ does not gain any advantage to select tuple (p_x, u_x) for interface $validate_{SA}$.
PH	✓	$\mathcal{A}$ observes $\vec{e_i}$, is given access to $\lvert U_f \rvert$, and can infer P_f. Yet, $\mathcal{A}$ does not gain any advantage to select tuple (p_x, u_x) for interface $validate_{PH}$.
SU	✓?	While $\mathcal{A}$ can easily pick p_x, p_y such that they map together, due to knowledge of P_f, she may not be able to map them to u_x. Thus, the notion may still be achieved, however, the probability for $\mathcal{A}$ to break it is higher.
WU	✓?	Same reasoning applies as for notion SU.
PS	✓	$\mathcal{A}$ observes $\vec{e_i}$ and is given access to (or, is able to infer) P_f. Yet, $\mathcal{A}$ does not gain any advantage to select tuple $(\vec{p}_{u_x}, u_x)$ for interface $validate_{PS}$.
AN	✓	$\mathcal{A}$ observes $\vec{e_i}$ and is given access to U_f and (and, is able to infer) P_f. Yet, $\mathcal{A}$ does not gain any advantage to select tuple $(\vec{p}_{u_x}, u_x)$ for interface $validate_{AN}$.
WA	✓	$\mathcal{A}$ observes $\vec{e_i}$ and is given access to U_f, Q_f and (and, is able to infer) P_f. Yet, $\mathcal{A}$ does not gain any advantage to select tuple $(\vec{p}_{u_x}, u_x)$ for interface $validate_{WA}$

4.7 Summary

Young experts (*i.e.* privacy researchers and academics) are struggling to analyse privacy notions and generate new insights and relationships between privacy notions due to the sheer amount of notions proposed in recent years. Non-experts are struggling, respectively, to find a solutions to argue, reason and compare privacy properties of their systems and third-party software. Predominantly, this can be traced back to a couple of factors: (a) the complexity of the privacy notions; (b) the sheer amount of different privacy notions (abstracting various aspects of privacy); (c) no common framework; and (d) a lack of automated tools for analysis and the generation of new insights.

In addition, privacy remains vaguely defined and can mean different things to different people; and, privacy is context-sensitive and application-dependent. Regulations, privacy principles and guidelines, and frameworks such as privacy by design take a first step towards improvement of above factors for experts and non-experts, but there is still a missing link of how to analyse and reason about privacy notions.

This chapter contributed towards resolving these issues: it presented an evaluation and analysis of the privacy games introduced in the previous chapter. First, our investigation introduced two classes of adversaries — a release adversary and an in/out adversary. The former class is limited to observe only the output of a system, while the latter is, in addition, able to inject elements and observe input elements. Next, we have identified some generic limitations of the privacy games that are inherent due to our system, adversary and privacy model. For each class of our privacy games, we stated the probability that an adversary is able to win an associated game — these probabilities help non-experts to obtain bounds for idealised versions of privacy-preserving systems.

As a first step towards automation, we present our software library of the previous introduced privacy games. This implementation may help to build further tools to

automatically analyse privacy-preserving applications and, consequently, contribute towards the generation of new insights and relationships between privacy notions. Mainly targeting non-experts as well as experienced privacy researchers, we continued to outline policies for the selection of a 'fitting' privacy notion or game. Following our guidelines, we aim to provide a better understanding of the syntactic privacy games for non-experts.

Having presented a theoretical analysis of the privacy notions and games, we demonstrated their applicability within a case study based on content-based clustering and collaborative filtering-based classification recommender systems. Via the case study we showed how the game-based privacy definitions have the potential to interconnect privacy implications and to reason about privacy properties within the context of recommender systems as well as any other systems that may be representable using our system model.

Overall, the contributions of this chapter aim to promote the understanding of existing privacy notions and introduces tools that may help to generate new insights for both, experts and non-experts. This chapter concludes Part I. In Part II, we derive requirements for large-scale multi-party applications via an analysis of various use cases. We focus on cases where data remains split over multiple mutually distrustful parties — in the context of privacy-preserving data analysis and data release. The following chapter utilises these constraints to propose an adapted approach from the trusted computing domain, and presents an advanced threat model and performance evaluation to prove the superiority of this approach in comparison to existing solutions.

Part II

Application to Practice

» *Your data, powering your experiences, controlled by you.* «

— Satya Nadella, CEO of Microsoft

5 REAL WORLD CONSIDERATIONS AND CONSTRAINTS

Contents

Publication Data

This chapter is based on the following publication:

(1) Robin Ankele, Kubilay A. Küçük, Andrew Martin, Andrew Simpson and Andrew Paverd. Applying the Trustworthy Remote Entity to Privacy-Preserving Multiparty Computation: Requirements and Criteria for Large-Scale Applications. In: Proceedings of ATC 2016. 2016. [34]

In this chapter, we elaborate on constraints arising from real world deployment of privacy-preserving analysis and release systems. We start by discussing four popular architectural design choices for distributed systems: the multi-party computation model, the federated learning model, the outsourced computation model, and a fully-split distributed model. To show the contrast to the real world, we introduce an idealised model compiled from requirements appearing in recent literature. Importantly, to identify real world constraints, we present a list of use cases pertaining to the setting of distributed multi-party applications. In such an environment, data typically remains split between mutually distrustful parties. Utilising the compiled list of use cases, we derive six requirements for large-scale prototypes. In this context, privacy-preserving systems are considered secure if they fulfill these requirements. Furthermore, in any real world environment it is important to recognise that performance — be it in terms of efficiency, utility, flexibility, scalability or the level of privacy — are the factors that typically count in the selection process of a software system. Our stated requirements additionally capture these constraints. Finally, we summarise the contributions of this chapter.

5.1 Overview

In Chapters 3-4, we discussed privacy considerations mainly from a theoretical viewpoint based on an 'idealised' world. In this idealised world model, academics typically make assumptions to model and abstract the 'real' world in order to develop their contributions. Such modelling is important, for example, to limit the attack surface of an adversary, to make assumptions about the behaviour of participants interacting with the system, or to construct and to analyse 'secure' primitives in limited settings. Prominent assumptions (in the cryptographic community) include: (a) a computational device is trusted to be secure; (b) an adversary has restricted or limited power; (c) an adversary follows certain behaviour or rules (*e.g.* honest, semi-honest or malicious); (d) hardness of computational problems (*e.g.* discrete logarithm problem, integer factorisation problem); (e) existence of idealised primitives (*e.g.* one-way functions, trapdoor functions); or (f) black box modelling. Further examples in this regard are discussed by Goldwasser and Kalai [197] and Naor [326].

While these assumptions hold in idealised world modelling, in the real world it is typically not possible to restrict the adversary and/or the environment (in which a system operates) to these assumptions. Some of these assumptions, *i.e.* (a), (e) and (f) may be appropriate for idealised systems. Others, such as (b) and (c), may be of direct importance in both real-world systems and modelling idealised systems — in giving an understanding where particular mechanisms are appropriate (and where they are not). Finally, assumptions such as, and similar to, (d), are fundamentally important to hold in both settings.

In real-world settings, some assumptions can not be guaranteed. For example, participants may deviate from their expected strategy due to changing incentives, security guarantees of 'theoretically secure' algorithms may be void due to errors in their implementation and trust is often based on historical behaviour of a system, but can change over time. To overcome such a divergence (between the ideal world and the real-world), the field of security economics [198, 16] studies various trade-offs between security, usability and efficiency.

In many cases the properties of utility, flexibility, efficiency and scalability trump privacy and security requirements of systems. It is in these environments in which real world applications operate. In the design of secure systems it is thus of utmost importance to recognise these limitations and tackle related issues.

The real world typically adds restrictions and limitations to the already long list of requirements in the design of secure systems. Not only does a system designer need to take into consideration the existing known attacks, but also potential new attack vectors that can and will evolve due to advances of technology and novel discoveries. The interplay between multiple disciplines complicates the already tricky quest to design secure systems — knowledge is wide reaching and ever extending.

The lack of commonly accepted notions for privacy are a limitation for systems to be designed in a provable secure manner. To tackle these shortcomings, it is essential to study a variety of use cases, which operate in an expected context, in order to derive a list of requirements from the real world. Further, from these findings, one can make an 'informed' analysis and state requirements and constraints for secure systems.

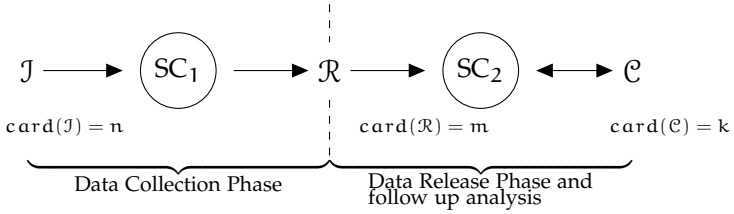


Figure 33: In a distributed data release mode the same multi-process approach as for singular data release is followed: multiple record owners ($\mathcal{I}$) provide their data to one (or potentially multiple) data curators ($\mathcal{R}$) in the first process (SC_1). In a separate process (SC_2) the collected information is provided for any further processing to multiple data analysts $\mathcal{C}$.

In this chapter, we analyse various real world use cases pertaining to the secure multi-party setting. From this analysis, we identify real world constraints and derive requirements for privacy-preserving large-scale applications. Our focus is on distributed systems in the context of privacy-preserving data analysis and data release. Thus, the first part of this chapter identifies different models in this context and presents features of an idealised model that our real world abstraction aims to follow closely.

5.2 Privacy Considerations in Distributed Systems

In Part II of Chapter 2 we introduced the centralised data release and data analysis paradigm, as well as systems operating in such settings. In this chapter, we extend our considerations to distributed environments. These kind of systems occur predominantly in the real world. The use cases discussed later in this chapter and the contributions in the following chapter are driven by distributed systems. In this context, the following distributed usage models can be seen as a foundation of the presented use cases; essentially, these models abstract the underlying building blocks of the use cases. The usage models and use cases, taken together, drive the developments of the next chapter.

5.2.1 Distributed Usage Models

In a distributed multi-entity environment, the architectural design choices may be different to those of a centralised system. Figure 33 illustrates the distributed data release mode, which follows the same design choices as the singular data release mode[1]. However, in Figures 34-37, we illustrate a classification of the most important architectural models based on the secure computation paradigms of Laud *et al.* [278], which follow the distributed data analysis mode.

[1]The *mode*, *behaviour* or *strategy* of a system describes the processes and behaviour that are invoked when a system is triggered by a signal (usually an input). The different modus operandi are defined in the design phase of a system. The following modes are based on the paradigms of Laud *et al.* [278].

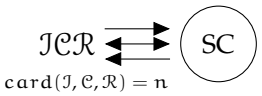


Figure 34: This mode follows the general multi-party computation model, where each of the participants acts as an input, computational and result player.

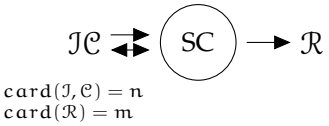


Figure 35: This mode follows the federated learning model, where the evaluation of the data analysis task is performed locally on the side of the input players and the result is transfered to the result player(s).

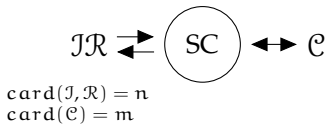


Figure 36: This mode follows the outsourcing of computation, where input and result players reside on the same entities, but the evaluation of the data analysis task is outsourced to computational player(s).

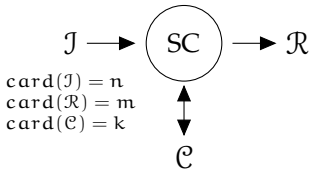


Figure 37: This mode follows the generic data analysis model (in a distributed fashion), where each player resides at a different entity

We follow the notation of Laud *et al.* [278] and denote entities with regards to their functionality. In both modes, input players (record owners) are denoted as $\mathcal{I}$. In the data release mode, we denote 'result players' (data curators) by $\mathcal{R}$ and computational players (data analysts) by $\mathcal{C}$. In the data analysis modes, we denote computational players (data curators) by $\mathcal{C}$, and result players (data analysts) by $\mathcal{R}$. The cardinality $\mathtt{card}(\cdot)$ of each of the players is given by a non-negative integer.

The mode in Figure 34 represents a general multi-party application, where each of the parties fulfills all roles by providing inputs to the data analysis evaluation, contributing to the processing and benefiting from the results of the processing. Examples of such applications are an average calculation of several parties' earnings without revealing the result to outsiders and multiple companies or agencies performing joint market research on the union of their datasets.

The mode in Figure 35 represents an application, where the input and computational players reside on the same party, but the result of the processing is provided to any external parties. Examples of such applications include companies or organisations which are interested in a data analysis result, which are however not willing or capable of performing the analysis themselves and therefore outsource such processing to a set of other entities. A popular example of this mode is federated learning [311], where a machine learning model is computed locally by the input players and the aggregated results are provided to the result players. Such a mode inherently provides privacy of the input players' data.

The mode in Figure 36 represents the task of outsourced computation of one's own data. The data analysis task is outsourced to a cloud provider, which takes the inputs of the party, performs the computation, and returns the result to the party. Examples of such applications include any data analysis operation on large datasets (*e.g.* calculating the average, max, min and sum over data).

The mode in Figure 37 represents the outsourced processing of collected data. This handles situations where the party should not have access to the processed data and therefore collects such data from other input players and the data analysis task is performed on outsourced computational players. The party of interest only benefits from the results of the data analysis operation. An example for such a mode is a medical research laboratory, which collects data from several hospitals and, due to limited computational resources or the large-scale nature of such data, outsources the computation to a high-performance data-processing centre.

As a disclaimer, we note that the modes presented are not exhaustive. Designers of privacy-preserving systems may introduce new modes and/or use other modes as described previously. Further, applications may explore different combinations of the presented modes and/or change the participating entities. The modes and examples presented serve as illustrations of some possible combinations.

5.2.2 An Idealised Model

An idealised model in the setting of large-scale and distributed multi-party applications includes various features to ensure secure and reliable operations. The following lists properties for the design of a secure system based on the synthesis of various contributions from the literature including [401, 400, 334, 335, 260, 317]. From an information security perspective, the following features may be required:

- ◇ Use of well analysed security primitives.
- ◇ Secure under the framework of provable security.
- ◇ Strong security bounds.
- ◇ Long-term security (*i.e.* post-quantum secure).
- ◇ Side-channel secure implementations (*e.g.* white-box cryptography).
- ◇ Secure under the framework of universal composability [93].
- ◇ Formal verification of privacy-preserving algorithms.
- ◇ Secure against honest-but-curious adversaries as well as malicious adversaries.

Furthermore, from the viewpoint of the effectiveness of such a system, the following features may be required.

- ◇ Efficient and scalable approaches (*i.e.* support for many participants).
- ◇ Communication and computational resource effectiveness (*i.e.* with regards to time and memory complexity).
- ◇ Convenient and application specific latency and throughput.
- ◇ Independent of the architectural design choices.
- ◇ Fault tolerance preventing accidental failures or adversarial attacks.

Finally, from a usability perspective, the following features may be required.

◇ Flexible and extendable to various applications.

◇ Utility of data values and operational results.

◇ Simplicity (*e.g.* easy to implement, to educate, understand).

◇ Applicability to various applications.

We note that the above listed features are neither complete nor widely applicable, but indicative and should, therefore, serve to clearly define research objectives and to validate and compare models. Some of the listed features are vaguely defined, without strict bounds or requirements such that they remain applicable to a wide field of privacy-preserving systems. Furthermore, these features remain open for change and can evolve if necessary to support and include new findings and cope with technological changes.

5.2.3 Our Settings

The setting in which our research is performed comprises the following. Parties are distributed, yet connected via a network (*e.g.* a local area network or the Internet). Parties supply and process their sensitive private input values. Therefore, the parties themselves are mutually distrustful in sharing their sensitive information with other parties that they have not established a trust relationship with. On the other hand, parties, however, are interested in processing a data-mining operation on the union of their data in order to gain knowledge about interesting patterns within their collective. Moreover, typically, we consider solutions for a large-scale setting. Large-scale in this context means either a large number of participating parties or the processing of large-scale data (which can further be distinguished between a small set of large data values or a large set of small data values).

5.3 Identifiying Real World Constraints

We have defined several use cases (network monitoring, billing and demand control in the smart grid; traffic monitoring and billing for road pricing schemes; genomics and intrusion detection in data analysis; auctions; election; and set intersections) for large-scale applications in the multi-party setting. All these use cases underly the generic systems of data analysis and data release as presented previously in this chapter and in Part II of Chapter 2. We split our use cases into two parts, on the basis of their underlying computational model — either a client-server model (*i.e.* many-to-one) or a MPC model (*i.e.* many-to-many). We note that a comprehensive survey of all applicable MPC use cases would be beyond the scope of this dissertation; rather, we focus on considering a representative sample of use cases to motivate and validate our research. Additionally, we have chosen to characterise each use case briefly, to avoid going down a rabbit hole (*e.g.* to create complex and chaotic use cases that are difficult

to understand and analyse). With the definition of the use cases we aim to identify real world constraints and, importantly, to derive requirements for large-scale applications (with a view to preserving the privacy of an individual or a group of individuals). In the following analysis, we are driven by properties such as efficiency, scalability, security (and privacy)[2], accuracy and flexibility; and other real world properties such as a small TCB[3] size, trust assumptions and applicability to a wide range of applications.

For each of the following use cases, we give a brief overview and for selected use cases we present pseudo code of possible implementations of such algorithms[4].

5.3.1 The Smart Grid

Overview. The smart grid operates on privacy-sensitive values and comprises a huge amount of mutually distrustful parties. MPC techniques can be used to manage energy resources alternatively in a more privacy-preserving way giving users self-control over their data and assurances of computations [351, 11]. We have identified the following three information flows of a modern smart grid: network monitoring, billing, and demand control. The first two follow the client-server model, whereas the third follows the MPC model.

References. Privacy and security considerations in the smart grid have been studied extensively in the literature. Our summary is mainly based on a technical report by the author [33] and Paverd *et al.* [350, 349, 352, 353, 351], but also considers [86, 42, 359, 99, 364, 251, 41].

Use Cases

Problem 1 (Network Monitoring)**.** *In order to generate and distribute energy resources in an efficient and reliable way, the energy provider has an interest in gaining as much information (e.g. time-of-use; energy usage) as possible from the energy consumer. Therefore, each consumer has to supply the energy provider with measurements of their energy consumption at certain time intervals (e.g. half-hourly; hourly; daily). However, these measurements contain privacy-sensitive information. Through non-intrusive load monitoring techniques [221], a potential malicious energy provider can detect patterns and artifacts in the overall energy consumption of various appliances and therefore diminish the privacy of the consumer.*

Analysis (Sensitive values)**.** *highly-time dependent energy measurements, anonymity of energy consumer*

Problem 2 (Billing)**.** *The smart grid enables the deployment of new billing schemes such as time-of-use [62] or dynamic billing [74]. While in the time-of-use scheme, the price varies*

[2] While it may seem that privacy is 'just' one factor in a long list and given the nature of this dissertation, we want to emphasise that (a) the other listed factors are often very relevant in feasibility / design decisions, and (b) the requirement of privacy is often 'inherently' required as a must (thus, in this context, not emphasised as a priority).

[3] The Trusted Computing Base (TCB) is the minimum software stack a user must trust.

[4] In our pseudo code, we focus on implementations based on MPC schemes and based on trusted computing schemes. Consequently, the code is influenced from such terminology. We recognise, though, that there are also other ways to preserve privacy in such use cases (*e.g.* by using cryptographic solutions).

during certain times of the day, in the dynamic billing scheme the price varies depending on the energy demand. For these schemes, the energy provider has to supply the consumer with the varying prices. However, the energy provider does not trust a potential malicious consumer to calculate their bill on their own. Mutually, the consumer does not trust the potential malicious provider to calculate the bill. Moreover, the protocol must protect against fraud of customer and disrupted or falsified pricing information.

Analysis (Sensitive values)**.** *time-sensitive billing information, anonymity of energy consumer, non-repudiation of energy consumer*

Problem 3 (Demand control)**.** *Demand control [353, 367] is a proactive solution to establish a reliable and robust smart grid for unpredictable consumer behaviour or unplanned drops or rises in energy consumption. In such a case, the energy provider broadcasts a request for reduction of energy to the consumers. Next, the provider and consumer interact in a bidding protocol to reduce the energy demand. However, in this protocol, the privacy of the bids and anonymity of the bidders should be guaranteed.*

Analysis (Sensitive values)**.** *privacy of bids, anonymity of bidder, non-repudiation of bidder*

Pseudo Code. Algorithm 1 presents the smart grid use cases. In the offline phase, each participant p_i obtains a unique identifier to identify itself as a participating party. In the online phase (network monitoring), every x minutes each participant p_i sends their energy consumption m_{t_x} at time t_x to the energy supplier. At any given time t_y an analyst p_x may query the system for the current energy consumption, $E[t_y]$. In the online phase (billing), every y minutes each participant p_i sends their energy consumption m_{t_y} at time t_y[5] to the energy supplier. At any given time t_z a participant p_x may query the system for their current bill, $\mathtt{bill}$. In the online phase (demand control), after receiving a request for demand control each participant p_i submits an energy reduction bid $p_i.\mathtt{bid}$ to the system and engages in an auction protocol. Consequently, the system analyses all bids and notifies all participants of the outcome of the auction (*i.e.* true or false).

Remark 3. *In the following, we present a threat model, systematic security analysis and security guarantees for the* network monitoring *use case of Algorithm 1. For the billing use case, we refer to Section 5.3.2, which discusses a similar use case for a road pricing scenario. Further, for the demand control use case, we refer to Section 5.3.4, which discusses a similar use case for auctions. While there may be different actors (and adversaries), the underlying threat model, security analysis and security guarantees apply in all cases.*

Security Analysis (Threat Model)**.** As adversary for the network monitoring use case, we assume a burglar, $\mathcal{A}$, whose aim is to learn the energy usage of a home owner, B. $\mathcal{A}$ may be monitoring a house she wants to break in and aims to learn information at what times B is at home. We assume that $\mathcal{A}$ can not influence B or directly gain access about B's private energy information. (In other words, we assume B can be abstracted as a black box.) We also assume that $\mathcal{A}$ may be monitoring multiple houses at the same time, thus, can't physically monitor all of them (at the same time). Further, we assume

[5] In order to preserve privacy, t_y may be much larger than t_x (traffic monitoring), thus a measurement m_{t_y} may not reveal patterns or artifacts of the appliances of the consumer.

Algorithm 1: Pseudocode for the smart grid use cases.

Precondition: Each participant p_i is a registered user[†].

Postcondition: Each p_i learns the current network usage, their bill, and if she won the auction for demand control.

Procedures:

proc $sendEnergyUsage(p_i, t_x, m_{t_x})$:
  $p_i.send(t_x, m_{t_x})$
proc $getEnergyUsage(t_y)$:
  for $i : 0..n$: $E[t_y] = E[t_y] + p_i.m_{t_y}$ **return** $E[t_y]$
proc $getBill(p_i)$:
  for $i : t_0..t_y$: $bill = bill + (p_i.m_{t_i} * price_{t_i})$ **return** $bill$
proc $sendBid(p_i, bid)$:
  $p_i.send(bid)$
proc $getWinnerAuction()$:
  for $i : 0..n$: **if** $p_i.bid > temp$ **then** $temp = p_i.bid$
  for $i : 0..n$: **if** $p_i.bid == temp$ **then** $p_i.notify(true)$ **else** $p_i.notify(false)$

Offline Phase:
Each p_i obtains a unique identifier to identify itself as a participant.

Online Phase (Network Monitoring):
invoke (every x minutes)
  for $i : 0..n$: $p_i.sendEnergyUsage(p_i, t_x, m_{t_x})$
$p_x.getEnergyUsage(t_y)$

Online Phase (Billing):
invoke (every y minutes)
  for $i : 0..n$: $p_i.sendEnergyUsage(p_i, t_y, m_{t_y})$
$p_x.getBill(p_i)$

Online Phase (Demand Control):
invoke (in case of demand control request)
  for $i : 0..n$: $p_i.sendBid(p_i, bid)$
$p_x.getWinnerAuction()$

[†] Energy authorities are able to bill and identify the user.

that B is registered with his energy provider and B's energy information is shared at a regular schedule (*e.g.* every 15 minutes). We assume that the energy provider evaluates the submitted energy information in a secure manner (confidential and integrity preserving) and has no incentive to record / report a wrong result (*i.e.* the energy provider acts independent and is not colluding with any other participating parties). The computations of the energy provider are performed in a black box manner.

Security Analysis (Systematic Evaluation)**.** Following Algorithm 1, we consider the following scenario:

1. Every x minutes B sends its private energy usage measurement (via $p_i.sendEnergyUsage(p_i, t_x, m_{t_x})$) to the energy provider via an agreed communication channel.

   Analysis:
   (a) the private energy usage must be kept confidential $\mapsto$[6]energy usage privacy;
   (b) the identity of the home owner must be confidential $\mapsto$ anonymity of home owner.

2. At any time, the home owner B must be able to request the total energy usage (via $p_i.\text{getEnergyUsage}(t_y)$) of the network.

   Analysis:
   (c) the reported results must not reveal any other home owners' information ↦ other home owners' energy usage privacy;
   (d) the reported results must be correct ↦ correctness of computation.

Security Analysis (Security Guarantees)**.** We evaluate Algorithm 1 in the following scenarios:

- **Idealised setting:** Under the assumption that each participant in the protocol is abstracted as a secure black box [245] and assuming that the participants communicate via secure channels [96, 93] then the protocol guarantees:
  (a) energy usage privacy: individuals energy usage information is not accessible to adversaries at participants-level (black box) and protected during communication (secure channel);
  (b) anonymity of home owner: the unique identifier is not accessible to adversaries at participants-level (black box) and protected during communication (secure channel);
  (c) other home owners' energy usage privacy: the reported total energy usage is protected during communication (secure channel) and doesn't contain any sensitive information (*e.g.* personally identifying information), but only the sum of all home owner's energy usage;
  (d) correctness of computation: the energy provider has no incentive to compute a wrong result (assumption).
- **Real-world setting:** Since black boxes and secure channels are idealised primitives (and do not exist in the real-world), to achieve the required security guarantees, privacy-preserving mechanisms must be deployed. In the context of this dissertation, we recommend the use of secure multi-party schemes and/or TREs (further explored in Chapter 6).

Complexity Analysis. Table 9 presents a complexity analysis of the procedures of Algorithm 1.

5.3.2 Road Pricing

Overview. New tolling and road pricing schemes need to capture and analyse a high load of privacy-sensitive values during certain times of the day (*e.g.* peak hours). Optimised MPC solutions using privacy-preserving analytics algorithms can be used to solve this problem. We have identified two information flows following the client-server model in the road pricing scenario: traffic monitoring and billing.

References. Our summary is informed by [233, 48, 250, 315, 419, 416].

[6] The symbol, ↦, in these contexts, should be read as 'gives rise to' or 'leads to' and should not be mistaken by its meaning as a mathematical symbol.

Table 9: Complexity Analysis of Algorithm 1. t represents the number of times that the energy usage is sent from a participant to the energy provider and n the number of house holds that are supplied by the energy provider.

Procedure	Complexity
Network Monitoring	
$\text{sendEnergyUsage}(p_i, t_x, m_{t_x}))$	$\mathcal{O}(t * n)^{\dagger}$
$\text{getEnergyUsage}(t_y)$	$\mathcal{O}(n)$
Billing	
$\text{sendEnergyUsage}(p_i, t_x, m_{t_x}))$	$\mathcal{O}(t * n)^{\dagger}$
$\text{getBill}(p_i)$	$\mathcal{O}(t)$
Demand Control	
$\text{sendBid}(p_i, \text{bid})$	$\mathcal{O}(n)^{\ddagger}$
$\text{getWinnerAuction}()$	$\mathcal{O}(2 * n)$

† The reported complexity includes the number of invocations (*i.e.* t times) of this function in the protocol.

‡ The reported complexity includes the number of invocations (*i.e.* n times) of this function in the protocol.

Use Cases

Problem 4 (Traffic Monitoring)**.** *Traffic authorities have an increasing interest in gathering traffic information and performing analysis on this data. There are several ways to collect this information; an efficient way is to detect passing vehicles using identifiers (e.g. licence plate, on-board-unit) at toll stations placed at certain road segments. The collected information can then be used to analyse road usage in order to detect and prevent traffic jams or congestion, to get statistics on road deterioration, and to get information on what type of vehicles (e.g. car, truck, motorcycle) are using the road. Furthermore, the results can be used in route planners. However, the collected information is privacy-sensitive with respect to real-time tracking and the detection of movement patterns of certain vehicles.*

Analysis (Sensitive values)**.** *location and time of road usage, anonymity of vehicle*

Problem 5 (Billing)**.** *Such a system enables new road billing schemes, where the vehicle driver can be charged on the basis of distance and time (e.g. rush hour) of road usage. Additionally, this includes pay-as-you-drive insurance schemes [419, 416] or vehicle and road taxes, which only charge the driver when the car is in use. In order to support these schemes, the vehicle driver must commit to the submission of detailed traffic information (e.g. location of vehicle, time of use) to the system. However, to be deployable, the system must be fraud-resistant and assure vehicle drivers' privacy.*

Analysis (Sensitive values)**.** *anonymity of vehicle, non-repudiation of vehicle*

Pseudo Code. Algorithm 2 represents the road pricing use cases. In the offline phase, each participant p_i obtains a unique identifier to identify itself as a participating party. In the online phase, every time a participant p_i accesses a new road segment, it increases the road segment counter $\texttt{road\_seg}_{ctr}$ at the new location and decrease the counter at the old location (in order to ensure traffic monitoring). Moreover, the billing counter $p_i.\texttt{billing}_{ctr}$ is increased when p_i accesses the road segment. At any

Algorithm 2: Pseudocode for the road pricing use cases.

Precondition: Each participant p_i is a registered user†.

Postcondition: Each p_i learns her travel expenses and traffic information.

Procedures:

proc $\mathtt{registerRoadSegment}(p_i, loc_{old}, loc_{new})$:

$p_i.billing_{ctr}++; road\_seg_{ctr}[loc_{old}]--; road\_seg_{ctr}[loc_{new}]++;$

proc $\mathtt{requestTrafficInfo}(loc)$:

return $road\_seg_{ctr}[loc]$

proc $\mathtt{requestBillingInfo}(p_i)$:

return $p_i.billing_{ctr} * price$

Offline Phase:

Each p_i obtains a unique identifier to identify itself as a participant.

Online Phase:

invoke (in case of accessing a new road segment)

$p_i.\mathtt{registerSegment}(p_i, loc_{old}, loc_{new})$

$p_i.\mathtt{requestTrafficInfo}(loc)$

$p_i.\mathtt{requestBillingInfo}(p_i)$

† Traffic authorities are able to bill and identify the user.

given time, a participant p_i is able to request a traffic information update on the road segment or to request travel expenses for the current trip.

Security Analysis (Threat Model)**.** As adversary in the road pricing use case, we assume an insurance company, $\mathcal{A}$, whose aim is to learn the movement patterns of an insuree, B. $\mathcal{A}$ is the insurance provider of B and wants to learn information how regular and at what times B travels to adjust B's insurance policy accordingly. We assume that $\mathcal{A}$ can not influence B or directly gain access to B's private location information. (In other words, we assume B can be abstracted as a black box.) Further, we assume that B is registered into a country-wide road pricing system and B's location information is shared whenever B enters a new road segment. We assume that the road pricing server evaluates the submitted location information in a secure manner (confidential and integrity preserving) and has no incentive to record / report a wrong result (*i.e.* the road pricing server acts independent and is not colluding with any of the participating parties). The computations of the road pricing server are performed in a black box manner.

Security Analysis (Systematic Evaluation)**.** Following Algorithm 2, we consider the following scenario:

1. Every time B enters a new road segment, B sends its private location information (via $p_i.\mathtt{registerRoadSegment}(p_i, loc_{old}, loc_{new})$) to the road pricing server via an agreed communication channel.

   Analysis:
   (a) the private location information must be kept confidential ↦ location privacy;
   (b) the identity of the driver must be confidential ↦ anonymity of driver;
   (c) the driver must not be able to retract his/her location info.
   ↦ non-repudiation of road segment access.

2. At any time, driver B must be able to request the current traffic information (via $p_i.\text{requestTrafficInfo}(\text{loc})$).

   ↳ **Analysis:**
   (d) the reported results must not reveal any other drivers location information ↦ location privacy;
   (e) the reported results must be correct ↦ correctness of computation.

3. At any time, driver B must be able to request its current bill (via $p_i.\text{requestBillingInfo}(p_i)$).

   ↳ **Analysis:**
   (f) the reported results must be correct ↦ correctness of computation.

Security Analysis (Security Guarantees)**.** We evaluate Algorithm 2 in the following scenarios:

- ◇ **Idealised setting:** Under the assumption that each participant in the protocol is abstracted as a secure black box [245] and assuming that the participants communicate via secure channels [96, 93] then the protocol guarantees:

  (a) location privacy: individuals location information is not accessible to adversaries at participants-level (black box) and protected during communication (secure channel);

  (b) anonymity of driver: the unique identifier is not accessible to adversaries at participants-level (black box) and protected during communication (secure channel);

  (c) non-repudiation of road segment access: once the driver enters a new road segment, the billing state of the road pricing system for the driver is irrevocable altered (protocol) and can't be changed (assumption);

  (d) location privacy: the reported traffic info per location is protected during communication (secure channel) and doesn't contain any sensitive information (*e.g.* location information, personally identifying information), but only a counter with the number of segment users;

  (e)+(f) correctness of computation: the road pricing server has no incentive to compute a wrong result (assumption).

- ◇ **Real-world setting:** Since black boxes and secure channels are idealised primitives (and do not exist in the real-world), to achieve the required security guarantees, privacy-preserving mechanisms must be deployed. In the context of this dissertation, we recommend the use of secure multi-party schemes and/or TREs (further explored in Chapter 6).

Complexity Analysis. Table 10 presents a complexity analysis of the procedures of Algorithm 2.

Table 10: Complexity Analysis of Algorithm 2.

Procedure	Complexity
Traffic Monitoring	
registerRoadSegment(p_i, loc_{old}, loc_{new})	$\mathcal{O}(1)$
requestTrafficInfo(loc)	$\mathcal{O}(1)$
Billing	
registerRoadSegment(p_i, loc_{old}, loc_{new})	$\mathcal{O}(1)$
requestBillingInfo(p_i)	$\mathcal{O}(1)$

5.3.3 (Privacy-Preserving) Data Analysis

Overview. Privacy-preserving data analysis is concerned with the problem that two or more parties want to discover and analyse patterns in their private databases. In a large-scale setting, this can involve either databases with many tuples per relation (*i.e.* entries per table), or a small amount of tuples, but with each entry consisting of large-scale values. By using trusted computing and MPC techniques, this problem can be solved in a privacy-preserving way. In the following, we identify two different scenarios of privacy-preserving data analysis following the MPC model.

References. Privacy-preserving data analysis has attracted a lot of attention from researchers and practitioners in recent years. As such, there are various use cases and applications covering this field. We've selected two emerging topics: genomics and intrusion detection. Our summary is informed by [415, 113, 428, 252] and private conversations with Kirk Rockett[7] for the genomics use case; and [347, 399, 354, 295] for the intrusion detection use case.

Use Cases

Problem 6 (Genomics). *To learn more about the outbreak and behaviour of diseases, databases from several research groups and hospitals can be combined to allow for a more extensive and global analysis. These databases might consist of two different sets of data types: genomics data sets, containing high-order genetic information (i.e. 3-4 million data points per entry [415]); and clinical data sets, containing various other identifying and privacy-sensitive data about the patient. Additionally, there are many constraints in designing such a system. First, there are several legal issues in the transfer of patient data between various countries. Second, several privacy-sensitive values can be found in both data types. For example, genetic data can reveal kinship to other persons[8]. Furthermore, patient data, especially when releasing information about the disease, includes sensitive data values about the ethnicity or the location (e.g. small village) of the patients.*

Analysis (Sensitive values). *anonymity of patients, privacy of location and ethnicity of patients*

[7] Kirk is a research manager for genome-wide genotyping experiments at the Wellcome Center for Human Genetics — https://www.well.ox.ac.uk/people/kirk-rocket

[8] For example, consider an insurance company learning this information and raising the insurance fees in case of the detection of a high likelihood of heritable diseases such as certain types of cancer.

Problem 7 (Intrusion detection)**.** *In the intrusion detection use case we represent three different scenarios. In the first scenario [354, 295], consider two or more companies — each of them having a database containing metrics of intrusion behaviour. In order to improve their intrusion detection systems they need to share data patterns relevant to hackers' behaviour. However, sharing these data values might reveal privacy-sensitive information on their own intrusion detection system. Hence, such a model must support data mining or machine learning operations while protecting each private database.*

For the second scenario, consider a company that has recorded the adversary's behaviour during a recent break-in. Clearly, it does not want to share this information, since it contains privacy-sensitive data of the company's system. However, to identify the adversary it must query several profile databases without revealing its content.

In the third scenario [399], employees of a company are monitored to detect if they leak any sensitive-information to outsiders (e.g. competitive companies). Thereby, the behaviour of the employee must be monitored in real-time and compared to the set of sensitive information. If the employee crosses a certain threshold, an alarm can be raised. However, such a system must respect innocent employees' privacy.

Analysis (Sensitive values)**.** *privacy of databases or sensitive data pattern, privacy of employee*

The previously mentioned problems all involve (in some way) the comparison of two or more data sets — typically pertaining to different parties and containing a set of collected user data (the 'target' data set) and a set of already classified data (containing certain metrics such as diseases and intrusion behaviour). Thus, in Algorithm 5 (set intersections) we focus on the more generic problem of identifying intersecting elements within datasets. However, we use data sets and scenarios pertaining to the genomics use case. (More details about Algorithm 5 are given in Section 5.3.6 (Set Intersections).)

5.3.4 Auctions

Overview. In an auction, participants typically publicly announce their bids for certain goods. Yet, such information might be privacy-sensitive, for example, by leaking information on the competitiveness of a company or by revealing information on personal assets. In this context, MPC techniques can prevent other parties from learning the private inputs of certain participants. The following use case follows the MPC model.

References. Auctions are a popular use case for secure multi-party schemes. The famous Danish sugar beet auction [70] in 2008 was the first large-scale and practical application of secure multi-party computation. Consequently, the use case has been widely studied in various other papers including [327, 237, 424, 175, 323, 138]. Our summary is informed by those references.

Algorithm 3: Pseudocode for the auction use case.

Precondition: Each participant p_i chooses a private bid $p_i.bid$.
Postcondition: The participant with the highest bid $p_i.bid$ wins.

Procedures:

proc $sendBid(p_i, bid)$:
  $p_i.send(bid)$
proc $getWinnerAuction()$:
  for $i : 0..n$: **if** $p_i.bid > temp^{\dagger}$ **then** $temp = p_i.bid$
  for $i : 0..n$: **if** $p_i.bid == temp$ **then** $p_i.notify(true)$ **else** $p_i.notify(false)$

Offline Phase:

Each p_i obtains a unique identifier to identify itself as a participant.

Online Phase:

Each p_i submits their private bid $p_i.bid$ for evaluation.
  for $i : 0..n$: $p_i.sendBid(p_i, bid)$
$p_x.getWinnerAuction()$

† We assume this value to be initialized to 0.

Use Cases

Problem 8 (Auctions)**.** *In order to select the winner of an auction, the auctioneer must learn the inputs of all participating parties. In the simplest way, participants publicly announce their bids. However, by doing so, other parties learn certain information about each of the participants. Additionally, often even the detection of participation in an auction can reveal sensitive information. A privacy-sensitive auction scheme, therefore, must prevent an adversary from gaining any of the above discussed knowledge and merely reveal the result of the auction (i.e. the winner of the auction) and obfuscate the participation and private inputs of each participant.*

Analysis (Sensitive values)**.** *privacy of bids, anonymity of bidder*

Pseudo Code. In Algorithm 3 we illustrate the auction use case. In the offline phase, each party p_i obtains a unique identifier to identify itself as a participating party. In the online phase, each p_i submits their bid $p_i.bid$ for evaluation and receives a notification if the outcome was positive or negative.

Security Analysis (Threat Model)**.** As adversary in the auction use case, we assume a company, $\mathcal{A}$, whose aim is to learn the private bid of another company, B. $\mathcal{A}$ may be a competing company that wants to learn information about the financial situation of company B. We assume that $\mathcal{A}$ can not influence company B or directly gain access to company B's private bid. (In other words, we assume company B can be abstracted as a black box.) We further assume that the auctioneer evaluates the submitted bids in a secure manner (confidential and integrity preserving) and has no incentive to report a wrong result (*i.e.* the auctioneer acts independent and is not colluding with any of the participating parties). The computations of the auctioneer are performed in a black box manner.

Security Analysis (Systematic Evaluation)**.** Following Algorithm 3, we consider the following scenario:

1. At some point during the auction schedule, B sends a private bid (via $p_i.\text{sendBid}(p_i, \text{bid})$) to the auctioneer via an agreed communication channel.

   Analysis:
   (a) the private bid must be kept confidential ↦ privacy of bids;
   (b) the identity of the bidder must be confidential ↦ anonymity of bidder.

2. After successful collection of all private bids from all participants, the auctioneer, p_x calculates the winner of the auction (via $p_x.\text{getWinnerAuction}()$) and informs the parties, p_i of the outcome (*i.e.* either negative or positive).

   Analysis:
   (c) the reported results must not reveal any individual bids ↦ privacy of winner;
   (d) the reported results must be correct ↦ correctness of computation.

Security Analysis (Security Guarantees)**.** We evaluate Algorithm 3 in the following scenarios:

- **Idealised setting:** Under the assumption that each participant in the protocol is abstracted as a secure black box [245] and assuming that the participants communicate via secure channels [96, 93] then the protocol guarantees:

  (a) privacy of bids: individual bids are not accessible to adversaries at participants-level (black box) and protected during communication (secure channel);

  (b) anonymity of bidder: the unique identifier is not accessible to adversaries at participants-level (black box) and protected during communication (secure channel);

  (c) privacy of winner: the auction outcome is protected during communication (secure channel) and doesn't contain any sensitive information (*e.g.* personal-identifying information);

  (d) correctness of computation: the auctioneer has no incentive to compute a wrong result (assumption)

- **Real-world setting:** Since black boxes and secure channels are idealised primitives (and do not exist in the real-world), to achieve the required security guarantees, privacy-preserving mechanisms must be deployed. In the context of this dissertation, we recommend the use of secure multi-party schemes and/or TREs (further explored in Chapter 6).

Complexity Analysis. Table 11 presents a complexity analysis of the procedures of Algorithm 3.

Table 11: Complexity Analysis of Algorithm 3. n represents the number of participants involved in the auction.

Procedure	Complexity
sendBid(p_i, bid)	$\mathcal{O}(n)^\dagger$
getWinnerAuction()	$\mathcal{O}(2 * n)$

† The reported complexity includes the number of invocations (*i.e.* n times) of this function in the protocol.

5.3.5 Elections

Overview. In an election scheme, participants submit their votes in favour of a particular candidate. These election schemes are often majority votes between certain candidates. The election process is typically operated by a trusted third party and performed in a secret manner (*i.e.* other participants can not learn the vote of individuals). In applying techniques of trusted computing and MPC, the trusted third party can be dropped. The following use case follows the MPC model.

References. Elections are similar to auctions; while the basic functionality of an auction scheme is abstracted by a max function, elections use extended functionality such as compare, count and max. Consequently, in the secure multi-party computation literature [228, 133, 56, 325, 61, 203] election use cases are similarly popular. In addition to the previous references, our summary is informed by [18, 380, 377].

Use Cases

Problem 9 (Voting)**.** *To select the majority of votes in an election scheme, the private votes of the participating parties have to be analysed. This analysis has to result in the correct output (e.g. majority) of the collected votes and must be performed in a way that preserves privacy and anonymity. However, voting typically includes various other restrictions such as only registered voters are allowed to cast a vote and each voter is only allowed to vote once. Moreover, to guarantee correctness it is required to have non-repudiation of votes, but also to guarantee anonymity of the voters. The challenging task is to apply all these requirements in the context of a large-scale application while guaranteeing strong assurances of anonymity and privacy.*

Analysis (Sensitive values)**.** *privacy of vote, non-repudiation of vote, anonymity of voter*

Pseudo Code. In Algorithm 4 we present a simplified voting use case. In the offline phase, each participant p_i obtains a unique identifier to identify itself as a participating party. In the online phase, each p_i submits their vote $p_i.c$ for evaluation and receives a notification of the candidate with the highest votes.

Security Analysis (Threat Model)**.** As adversary in the elections use case, we assume a curious neighbour, $\mathcal{A}$, whose aim is to learn the private vote of another voter, B. $\mathcal{A}$ may be a supporter of one political party and wants to learn information about their neighbour's voting preferences (and may discriminate them according to the outcome). We assume that $\mathcal{A}$ can not influence voter B or directly gain access to voter

Algorithm 4: Pseudocode for the election use case.

Precondition: Each participant p_i privately chooses a candidate c_i from the candidate list $CandidateList$.

Postcondition: The candidate with the highest votes wins.

Procedures:

proc $castVote(p_i, c)^{\dagger}$:
  $CandidateList[c]++$
proc $determineWinnerElection()$:
  $c_{win} = max(CandidateList)$
  for $i : 0..n$: $p_i.notify(c_{win})$

Offline Phase:

Each p_i obtains a unique identifier to identify itself as a participant.

Online Phase:

Each p_i submits their private vote $p_i.c$ for evaluation.
  for $i : 0..n$: $p_i.castVote(p_i, c)$
$p_x.determineWinnerElection()$

† We assume that a participant is only able to cast a vote once per election.

B's private vote. (In other words, we assume voter B can be abstracted as a black box.) We further assume that the election authority (*i.e.* the party who counts the collected votes) evaluates the submitted votes in a secure manner (confidential and integrity preserving) and has no incentive to report a wrong result (*i.e.* the election authority acts independent and is not colluding with any of the participating parties). The computations of the election authority are performed in a black box manner.

Security Analysis (Systematic Evaluation)**.** Following Algorithm 4, we consider the following scenario:

1. At some point during the election schedule, B sends a private vote (via $p_i.castVote(p_i, c)$) to the election authority via an agreed communication channel.

   Analysis:
   (a) the private vote needs to be kept confidential ↦ privacy of vote;
   (b) the identity of the bidder must be confidential ↦ anonymity of voter;
   (c) the voter must not retract his/her vote ↦ non-repudiation of vote.

2. After successful collection of all private votes from all participants, the election authority, p_x calculates the winner of the election (via $p_x.determineWinnerElection()$) and informs the parties, p_i of the outcome (*i.e.* winning candidate).

   Analysis:
   (d) the reported result must not reveal any individual votes ↦ release privacy;
   (e) the reported results must be correct ↦ correctness of computation.

Table 12: Complexity Analysis of Algorithm 4. n represents the number of participants involved in the election and c the number of candidates standing for election.

Procedure	Complexity
castVote(p_i, c)	$\mathcal{O}(n)^\dagger$
determineWinnerElection()	$\mathcal{O}(c+n)$

† The reported complexity includes the number of invocations (*i.e.* n times) of this function in the protocol.

Security Analysis (Security Guarantees)**.** We evaluate Algorithm 4 in the following scenarios:

- **Idealised setting:** Under the assumption that each participant in the protocol is abstracted as a secure black box [245] and assuming that the participants communicate via secure channels [96, 93] then the protocol guarantees:
  (a) privacy of votes: individual votes are not accessible to adversaries at participants-level (black box) and protected during communication (secure channel);
  (b) anonymity of voter: the unique identifier is not accessible to adversaries at participants-level (black box) and protected during communication (secure channel);
  (c) non-repudiation of vote: once the vote is casted, the counting state of the election authority is irrevocable altered and can't be changed (assumption);
  (d) release privacy: the released result does not contain individual votes, but only a ranked candidate list (according to the votes);
  (e) correctness of computation: the election authority has no incentive to compute a wrong result (assumption).
- **Real-world setting:** Since black boxes and secure channels are idealised primitives (and do not exist in the real-world), to achieve the required security guarantees, privacy-preserving mechanisms must be deployed. In the context of this dissertation, we recommend the use of secure multi-party schemes and/or TREs (further explored in Chapter 6).

Complexity Analysis. Table 12 presents a complexity analysis of the procedures of Algorithm 4.

5.3.6 Set Intersections.

Overview. The problem of set intersections is one of the classical MPC problems[9]. A variant of the set intersections problem is the set membership problem. While for set intersections the number of intersecting elements are determined, for set membership the algorithm terminates when a target record is found in the data set. Techniques of trusted computing and MPC can help to guarantee an individual's privacy in such an environment.

[9]Similarily, see Yao's Millionaires Problem [447].

Algorithm 5: Pseudocode for the set intersection use cases.

Precondition: Each of the two participants p_1, p_2 hold a dataset $p_1.set, p_2.set$ of n-tuples.

Postcondition: p_1, p_2 learn the overlapping elements of $p_1.set \cap p_2.set$.

Procedures:

proc $sendSet(p_i, set)$:
  $p_i.send(set)$
proc $analyseDataset()$:
  forall p **in** $p_1.set$:
    forall q **in** $p_2.set$: **if** $p = q$ **then** $p_1.notify(p); p_2.notify(q); return$ •

Offline Phase:

for $i : 0..n$:
- $p_1.set \xleftarrow{\$} \mathcal{R}$, $p_2.set \xleftarrow{\$} \mathcal{R}$ ⋆ where $\mathcal{R} = [0, .., 100]$.
- $p_1.set \leftarrow \mathcal{R}$, $p_2.set \leftarrow \mathcal{R}$ ‡ where $\mathcal{R} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}^m$ with $m = 10k$, and at a (random)† position we insert the SNP $= \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.

Each p_i obtains a unique identifier to identify itself as a participant.

Online Phase:

Each p_i submits their private set $p_i.set$ for evaluation.
  for $i : 0..n$: $p_i.sendSet(p_i, set)$
$p_x.analyseDataset()$

• The function only (immediately) terminates for set membership testing.

⋆ For scenario 1.

‡ For scenario 2.

† For our test purposes, we choose 10 random positions in order to get matches.

References. Privacy-preserving set operations such as set union, set intersection and set membership have been widely studied utilising various technologies. For example, recently, zero-knowledge proofs are used to prove set membership without revealing individual values [58, 324]. Our summary is mainly informed by set operations in the multi-party setting, studied in [269, 182, 224, 118, 181].

Use Cases

Problem 10 (Set Intersections)**.** *The problem states that two or more parties want to determine the amount of intersection between their private databases. Typically their databases contain privacy-sensitive values; as such, it is not possible to just share the data and perform analysis on the joint database. A solution must reveal the number of intersecting elements and/or the overlapping values; nonetheless, the remainder of the data values in each database must not leak any information.*

Analysis (Sensitive values)**.** *privacy of database values*

Pseudo Code. Algorithm 5 presents two scenarios of set intersections with respect to genomic data. In the offline phase, each participant p_i obtains a unique identifier to identify itself as a participating party. For the first scenario, consider two hospitals

that want to compare their patient datasets [269, 328] and see how many overlapping patient they have, which reflects our first setting (high-order of tuples, with single data values). For the second scenario, consider two research labs, both holding a private dataset on genomic data. One of the datasets holds genomes with SNPs[10] related to patients with a certain disease; the other dataset holds genomes of patients. We compare these two datasets to see the likelihood that one of the patients in the second dataset could develop such a disease. Thus, in the offline phase, the datasets are filled with synthetic data values, *i.e.* random values from the range [0,100] for scenario 1 (the random numbers may refer to patient ids); and tuples of the form $\begin{bmatrix}0\\0\end{bmatrix}^m$, with $m = 10k$ and at a random position we insert a SNP $\begin{bmatrix}0\\1\end{bmatrix}$ for scenario 2. In the online phase, an analyst p_x may call at any given time the evaluation function, which, in turn, evaluates the data sets for intersecting elements and notifies both parties p_1, p_2 in case of a hit.

Security Analysis (Threat Model)**.** As adversary in the set intersections use case, we assume an insurance company, $\mathcal{A}$, whose aim is to learn the medical records of an insuree, B, whose data is included in the medical data set of a hospital. $\mathcal{A}$ is the insurance provider of B and wants to learn information about the insuree's health status in order to adjust the pricing of their insurance policy. We assume that two hospitals H_1 and H_2 are engaging in research activities and, therefore, sharing their patients' records (including B's health records) to enhance their research outcomes. In this context, we assume that $\mathcal{A}$ can not influence insuree B or directly gain access to insuree B's private health record from the hospitals H_1 and H_2. (In other words, we assume insuree B and hospitals H_1 and H_2 can be abstracted as black boxes.) The hospitals use a data processor, p_x, (*i.e.* the party who calculates the set intersection of both sets) who evaluates the submitted sets in a secure manner (confidential and integrity preserving) and has no incentive to report a wrong result (*i.e.* the data processor acts independent and is not colluding with any of the participating parties). The computations of the data processor are performed in a black box manner.

Security Analysis (Systematic Evaluation)**.** Following Algorithm 5, we consider the following scenario:

1. In order to enhance their research activities, at some point hospital H_1 and H_2 send their patients records (including B's record) (via $p_i.\text{sendSet}(p_i, \text{set})$) to the data processor via an agreed communication channel.

   ↳ **Analysis:**
   (a) the private health records must be kept confidential ↦ privacy of health records;
   (b) the patient associated with the health record must be anonymous ↦ anonymity of the patient;

[10] A Single Nucleotide Polymorphism (SNP) [415], is a variation of a single nucleotide that occurs at a specific position and deviates at exactly one nucleotide.

Table 13: Complexity Analysis of Algorithm 5. n represents the number of hospitals involved in the protocol, ℓ_1 the length of the dataset of Hospital 1 and ℓ_2 the length of the dataset of Hospital 2.

Procedure	Complexity
$\text{sendSet}(p_i, \text{set})$	$\mathcal{O}(n)^\dagger$
$\text{analyseDataset}()$	$\mathcal{O}(\ell_1 * \ell_2)$

† The reported complexity includes the number of invocations (*i.e.* n times) of this function in the protocol.

2. Once the data processor received both datasets, p_x evaluates the both sets (via $p_x.\text{analyseDataset}()$) and informs the hospitals, H_i of the outcome (*i.e.* overlapping records).

Analysis:
(c) the reported results must not reveal any individual health records $\mapsto$ release privacy;
(d) the reported results must be correct $\mapsto$ correctness of computation.

Security Analysis (Security Guarantees)**.** We evaluate Algorithm 5 in the following scenarios:

◇ **Idealised setting:** Under the assumption that both hospitals and the data processor can be abstracted as a secure black box [245] and assuming that the participants communicate via secure channels [96, 93] then the protocol guarantees:

(a) + (b) privacy of health records + anonymity of patient: individual health records are not accessible to adversaries at participants-level (black box) and protected during communication (secure channel);

(c) release privacy: the released results are protected during communication (secure channel) and only matching elements are released to the hospitals (protocol). We assume that both hospitals are allowed to learn the results (assumption);

(d) correctness of computation: the data processor has no incentive to compute a wrong result (assumption).

◇ **Real-world setting:** Since black boxes and secure channels are idealised primitives (and do not exist in the real-world), to achieve the required security guarantees, privacy-preserving mechanisms must be deployed. In the context of this dissertation, we recommend the use of secure multi-party schemes and/or TREs (further explored in Chapter 6).

Complexity Analysis. Table 13 presents a complexity analysis of the procedures of Algorithm 5.

Table 14: Overview of our analysis of the use cases, which are used to drive and validate our research. We use the generalised results to derive requirements for applications in large-scale and distributed multi-party environments.

Use Case	Composition	Sensitive Attributes	Complexity
Smart Grid			high
Network Monitoring	count + sum	time	
Billing	sum	sum. value	
Demand Control	min	bid	
Road Pricing			high
Traffic Monitoring	count	location, time	
Billing	sum	sum. value	
Auctions	max	bid	low
Elections	compare + count + max	vote	med
Set Intersections			med
Set Membership	compare	data entry	
Set Intersections	compare + count	data entry	
Genomics			high
DNA Sequencing	compare	DNA string	
Disease Susceptibility	compare	DNA string	

5.4 Analysis of our Use Cases

After having defined several use cases in a distributed multi-party setting, we now generalise and analyse these use cases to identify real world requirements. Following our analysis, we derive requirements from the generalised use cases.

We break the use cases down to simplistic operations[11] (*e.g.* max, compare, count, sum), indicate sensitive attributes and rate them according to the complexity of the use case. Table 14 presents a summary of our findings.

(a) **Smart Grid.** The network monitoring use case can be realised by performing count and sum operations over the submitted energy measurements, the billing use case by a sum operation over the submitted energy measurements and the price at the given time period, and the demand control use case can be realised by a minimum calculation, with regards to the price discount over the submitted bids. The committed energy values in traffic monitoring are highly time-sensitive, while the summation value and anonymity in the billing phase can be sensitive, and the bid and participation in the bidding protocol may be sensitive.

(b) **Road Pricing.** The traffic monitoring use case can be realised by a counter for each road segment (increasing if a road user enters the road segment and decreasing when leaving). In order to determine the bill for a road user, the number of road segments they travelled on must be saved. As such, the billing use case can be realised by summing the road segments. Sensitive values are the location and

[11] We recognise that each use case may involve multiple other operations to realise its tasks. With the indicated 'simplistic' operation (of each use case), we mean to highlight the basic data analysis operation, which the use case is relying on (in other words, the most prominent operation or the operation that forms, shapes or characterises the use case mostly).

time of road usage for each user and the summation value and anonymity at the billing phase.

(c) **Auctions.** The auction use case is represented by a maximum calculation over all bids. Sensitive values include the committed bid itself, and meta-data such as participation in the auction and number of bidding attempts.

(d) **Elections.** The election use case can be realised by counters for each candidate (increasing for each vote); consequently, the winner is determined by calculating the maximum over these counters. Sensitive values include the vote itself, the anonymity of the vote, and the non-repudiation of a committed vote.

(e) **Set Intersections.** In the set intersection use case, set membership tests are realised by a comparison operation and running through the dataset exactly once (the operation terminates in case of a hit), while for set intersection a comparison and a count operation is used; moreover, both (or multiple) datasets must be compared. The dataset values itself, such as the result of the testing can be sensitive values. In our case (using genomic data) the DNA strands are sensitive as well as meta-data such as pertaining patients data are sensitive.

5.5 Requirements for the Design of Large-Scale Prototypes

We now derive requirements from the generalised results of the analysis of the use cases. Overall, each previously stated use case handles privacy-sensitive data values of several data owners. The goal of these use cases is to collaboratively evaluate a function on the data pertaining to mutually distrustful parties. As such, our first requirement derives from this problem:

Requirement 1 (Privacy-Preservation)**.** *A system is required to* preserve *the privacy of* input *and* output *values. The former refers to* input privacy *and the latter to* release privacy*.*

A system that fulfills the input privacy requirement handles the input values in a confidentiality- and integrity-preserving manner and does not leak any information of the private inputs. Moreover, a system fulfilling the release privacy requirement does not leak any information of the private inputs from the function evaluation.

Often even just the participation in a collaborative function evaluation leaks information (*e.g.* participation in an auction of a high-value target) about individuals. On the other hand, non-repudiation of committed values (*e.g.* votes in an election) is desired. As such, as our second requirement we require the following:

Requirement 2 (Anonymity and Non-Repudiation)**.** *A system is required to preserve the* anonymity *of parties participating in the protocol. Moreover, for committed data values the system is required to provide* non-repudiation *of the committed values.*

In addition to the security and privacy requirements, the function evaluation must result in the correct output, even in the presence of an adversary. Preferably, such a

requirement should hold even in the case of a majority of malicious parties (consequently, such a system should be independent of the number of malicious parties).

Requirement 3 (Correctness)**.** *A system is required to provide* correctness *of the function evaluation; this should hold even in the presence of malicious parties.*

Following the *correctness* requirement, we further harden systems by requiring guaranteed output delivery of the result of the computational function evaluation, even in the presence of malicious parties.

Requirement 4 (Guaranteed Output Delivery)**.** *A system is required to guarantee* output delivery *of the result of the function evaluation to all, be they trustworthy or malicious.*

To refine the requirement of *release privacy* (for example, to provide a more accurate or efficient evaluation), we require the utilisation of a *privacy budget*. A privacy budget regulates the level of private information, which may be leaked by the output of the computational function. Such a 'budget' should be user-configurable and improve the accuracy of certain function evaluations[12].

Requirement 5 (Regulation of Privacy Level)**.** *Each party should be able to maintain a* privacy budget*; such a budget indicates the level of trust (of a user) in the application.*

In order to be deployable for real world applications, systems must execute their operations within a 'reasonable' time period. There are many factors influencing the execution time of a protocol. Examples include: the computational complexity of the function, the number of participants and the privacy/security requirements. This requirement is typically critical in the decision of the selection of the underlying protocol or application.

Requirement 6 (Efficiency and Scalability)**.** *A system is required to provide an* efficient *and* scalable *implementation of the privacy-preserving multi-party protocol (within the given constraints).*

Applications in the context of large-scale distributed environments require certain 'features' to be deemed secure. Fulfilling the requirements listed in this section is a good starting point for any privacy-preserving real world application. As such, our requirements aim to remain generic to cover a wide range of applications, yet, to be concrete enough to 'enforce' a certain standard. Moreover, we acknowledge that applications, partly or not at all fulfilling our listed requirements, may still be deemed 'secure'. To be, and remain secure, is mainly a point of designing 'secure' systems (as well as assumptions and realisations of the adversary model, system model and privacy model). Thus, our requirements serve as a way to achieve secure designs (in other words, we propose guidelines for secure designs), but should not be seen as the *only* way to achieve such a goal. What is more, most security and efficiency

[12] For example, a user Alice, less concerned about the privacy-leakage of her inputs, can improve the accuracy of the output of the computational function by defining a lower privacy budget value.

requirements are highly context-dependent — as such, different applications scenarios may have different requirements. Consequently, we state the following definition:

Definition 52 (Secure and Privacy-Preserving real world Systems)**.** *We consider a system to be 'secure'*[13] *if it fully, or partly (dependent on the application context), fulfills requirements 1-6.*

5.6 Summary

The 'ideal' world model is an limited abstraction of the real world. Typically, assumptions are made to design secure systems by restricting adversaries in their powers, assuming they follow given rules or strategies; computational devices are assumed to be trusted; and idealised primitives are assumed to exist.

In the real world, those restrictions and limitations do not apply[14]. As a consequence, designing secure systems in real world settings is hard. It is a complex task involving multiple fields and various actors. In this chapter, we focused on bridging this gap for privacy-preserving data analysis and data release. In this context, we focused on distributed models, formerly introduced by Laud *et al.* [278]: the multi-party computation model, the federated learning model, the outsourced computation model and a fully-split distributed model. These models represent different architectural design choices complementary to the centralised model.

An idealised model informs a real world abstraction with desired properties. In this chapter, we synthesised an idealised model from various contributions of the literature and stated features from an information security perspective, an efficiency perspective and an utility perspective. It is of utmost importance to recognise that properties such as efficiency, utility, flexibility and scalability are the driving factors in the real world; privacy and security properties are typically considered with a trade-off to the previous factors.

The use cases that we elaborate on are defined either in a client-server model (*i.e.* many-to-one) or a MPC model (*i.e.* many-to-many). We've defined and analysed various use cases: (a) the smart grid; (b) road pricing; (c) data analysis; (d) auctions; (e) elections; and (f) set intersections. Utilising this widespread collection we were in the position to derive requirements for large-scale prototypes.

To establish secure and privacy-preserving real world systems we defined the following requirements[15]: privacy-preservation, anonymity and non-repudiation, correctness, guaranteed output delivery, regulation of privacy levels and efficiency and scal-

[13] Under the assumptions of a real world environment, achieving 100% security is unlikely (in some cases even impossible to achieve). Typically, systems are secure up to a point, beyond which it is not profitable for an adversary to attack said system. Various trade-offs between security, usability and efficiency are studied in the field of security economics [198, 16].

[14] In the real world such restrictions are often enforced by acts of determent such as laws stating consequences if someone does not follow the 'rules'. Yet, this does not limit an adversary from exploiting the vulnerabilities and break an otherwise secure system. He/she just needs to live with the consequences in the case of getting caught.

[15] Designing a secure system is highly context-dependent. The stated requirements abstract a subset of possible avenues to create secure systems. As such, these requirements should not be taken as an ultimate set, but more as a 'minimum' set of requirements in the process of designing secure systems.

ability. Importantly, the requirements stated are kept generic to cover a wide range of applications.

System designers may utilise our list of requirements and constraints to validate their systems to satisfy secure, privacy-preserving and efficient design constraints. Our stated requirements may be seen as a subset of principles to satisfy. Still these set of requirements is not complete since security and privacy are broadly defined (as we established in our discussions in Part I of Chapter 2) and it's generally impossible to capture all aspects of all applications.

In the next chapter, we focus on the problem of input privacy. In this context, we propose an adapted approach from the trusted computing domain that attains superiority in terms of efficiency, scalability and flexibility with regards to existing solutions. We also discuss an advanced threat model addressing attacks on privacy models, but predominantly focuses on real world attacks such as side channel attacks. A performance evaluation of our approach in comparison to existing solutions is also presented.

» Those who would give up essential liberty to purchase a little temporary safety deserve neither liberty nor safety. «

— Benjamin Franklin, Polymath and Founding Father

6 EVALUATION AND ANALYSIS OF TRUSTWORTHY REMOTE ENTITIES IN DISTRIBUTED SETTINGS

Contents

Publication Data

This chapter is based on the following publications:

(1) Robin Ankele and Andrew Simpson. On the Performance of a Trustworthy Remote Entity in Comparison to Secure Multi-Party Computation. In: Proceedings of WCSF 2017. 2017. [31]

In this chapter, we elaborate on constraints arising from the real-world deployment of privacy-preserving analysis and release systems. We start by discussing the secure *multi-party computation* (MPC) paradigm, which aims to tackle the problem of *input* privacy. In our context, we present weaknesses of this approach. Next, we introduce a complementary approach based on trusted computing principles: Trustworthy Remote Entity (TRE) systems. In its simplest form, a TRE operates as an intermediary between participating communication parties. It portrays a trusted third party (TTP), but, contrary to a TTP that requires users to blindly trust all performed actions, a TRE can verify its trustworthy nature via a process called software attestation. As such, this approach may be used to operate on applications in the MPC domain in a scalable and efficient manner. Importantly, in this context, we present a threat model and security analysis covering attacks against privacy mechanisms as well as 'real-world' attacks targeting the implementation of applications such as side channel attacks. To

establish a fair comparison of our adapted approach, we evaluate it against existing secure two-party and multi-party computation systems. In this context, we test the relative efficiency of the two approaches via prototyping. Moreover, we critically review both approaches side-by-side and present advantages and disadvantages of the same. Finally, we summarise the contributions of this chapter.

6.1 Overview

In order to preserve an individual's or a group's privacy, system designers typically make assumptions[1] about the system model, privacy model and adversary model. Examples include: adversaries behave in certain ways (*e.g.* follow/deviate from the instructions of a protocol), adversaries have access to a certain subset of information (*e.g.* access to a limited number of interfaces), privacy is defined over a subset of certain properties (*e.g.* a privacy notion defines a goal; a privacy game defines steps to break the associated notion), and systems, which abstract a wide range of applications, are defined via a (limited) system model. In addition, previously described privacy models typically assume data to be shared with a trusted data curator, who holds a single dataset. (Users share their private input data with such a data curator.) In this context, an analyst receives either a 'sanitised' version of the dataset or is given query access to the dataset via an interface.

As we have already mentioned in the previous chapter, some of these 'idealistic' assumptions do not necessarily abstract real-world behaviour. Nowadays, computational tasks are seldom done in isolation, but rather involve distributed tasks executed in multi-party settings. Consequently, the assumptions of a trusted third party model are obsolete. Data is, and often remains[2], distributed over multiple parties. In such a distributed setting, the problem of *input* privacy (or *communication* privacy) arises: input values pertaining to a finite set of distributed users need to be communicated among the members (for example, to evaluate a function over all inputs) in a privacy-preserving fashion.

Communication privacy comprises certain cryptographic goals: integrity, authenticity, freshness and confidentiality. To achieve communication privacy, it requires *integrity* of the messages, in order to validate if a message has been tampered with during communication. It requires *authenticity*, in order to verify the identity of the communication partners. It requires *freshness* of messages, such that an adversary is not able to 'replay' a message which was already sent at an earlier time. Finally, it requires *confidentiality* of the messages, in order to ensure that only legitimate communication partners can learn the content of a message.

A common way to achieve these goals is the utilisation of an authenticated encryption scheme [54, 255, 375]. By including a counter in the messages (for freshness) and assuming only parties with the secret key can create legitimate messages (for authenticity), an authenticated encryption scheme guarantees confidentiality (via encryption)

[1] Similarly, we made such assumptions in the chapters pertaining to Part I.

[2] With remains, we understand that data, that is shared, is only an abstraction of the 'real' data. For example, secret shares [395] split a secret into several parts and the secret can only be recovered if one is in possession of a certain threshold of shares.

and integrity (typically, via a MAC[3] construction). Nevertheless, authenticated encryption schemes are typically defined for use in two-party communication and, as such, may create an overhead in the communication of multiple parties.

Alternative approaches have been defined: for example, *secure multi-party computation* techniques. Secure multi-party computation (MPC) allows multiple parties to jointly compute a function on their private inputs, without any of the parties learning anything about the inputs of other participating parties, except what can be determined from the output of the computation. Unfortunately, traditional MPC algorithms [301, 226, 236, 56, 67, 135, 89, 136, 169] are inefficient in the presence of a large number of participants or when performing a large number of (the same) operation(s)[4]. Moreover, traditional MPC schemes do not consider the problem of *release privacy* — which we very much focus on in this dissertation.

In this chapter, we critically analyse the cryptographic approach of MPC. In our opinion, current cryptographic approaches to tackle the MPC problem are preliminary and not sufficiently mature to solve generic large-scale MPC problems with the given constraints for such applications (*e.g.* latency, throughput, level of privacy) [279, 38, 432, 341, 155].

As a consequence, we adapt and utilise a different approach based on trusted computing principles: Trustworthy Remote Entities (TREs). TREs [351] use software attestation to verify their trustworthy nature. What is more, our TRE approach is based on the novel Software Guard Extensions (SGX)[5] by Intel, which promote secure, efficient and scalable processing for large-scale applications. Further, we present an in-depth threat model and security analysis of our approach as well as a detailed performance evaluation.

A recent review survey by Choi and Butler [114] on the historical improvements of multi-party schemes in comparison to trusted hardware confirms our findings. The authors conclude with three open challenges (partly addressed in this dissertation): (a) defeating malicious adversaries, (b) low-resource device friendly and trusted execution environment supported multi-party schemes, and (c) trusted hardware and privacy-preserving computation.

6.2 Analysis and Weaknesses of the Cryptographic Approach

In this dissertation, we are in particular interested in protocols that operate in large-scale settings and solve 'generic' problems (*i.e.* are applicable to many computational problems). Thus, it is paramount to optimise the protocols to run in an expected time and/or operate under restricted memory space limits. Nevertheless, the computational and communication overheads of MPC protocols are large and still magnitudes

[3] Message Authentication Code (MAC)

[4] The overhead mainly comes from transferring data from the plaintext domain into the MPC domain (and backwards). *E.g.* consider homomorphic encryption and computations with DNA data, where each data entry must be encrypted before performing any computations.

[5] https://software.intel.com/en-us/sgx

higher with respect to performing the collaborative computations in the clear. The following presents weaknesses[6] of MPC schemes. Our analysis comprises [57, 227, 301, 226, 236, 56, 67, 135, 89, 136, 169, 139, 142, 73, 381, 121, 190, 196, 286, 272, 290].

6.2.1 Communication Overhead

Many proposed protocols have large computational and communication complexity (*e.g.* $\mathcal{O}(n^8)$ for an implementation of the BGW protocol [57, 227], where n represents the number of participating parties). In particular, protocols that protect against malicious adversaries [223, 114] suffer from high communication complexity mainly due to sophisticated techniques to achieve resilience against faults. Such protocols are typically very communication-intensive. Many protocols depend on factors such as: the circuit size s; the number of participants n; the security parameter λ; or the rounds of communication d [139]. To establish efficient protocols for large-scale implementations, optimally these protocols should be independent of any or most of these factors. As a consequence, Damgård and Nielsen [142] proposed protocols that reduce communication complexity to linear complexity for passive security and to quadratic complexity for active security. Countermeasures to tackle the problem of communication complexity are techniques such as: communication locality [73], load balancing [73] or utilising so-called quorums [381].

6.2.2 Computational Overhead (FHE)

Fully homomorphic encryption-based MPC schemes propose powerful and useful features, but face a significant computational overhead compared to conventional MPC approaches and/or conventional encryption schemes. To achieve the same (or, at least, similar) guarantees of security the performance of the scheme is limited. Some generic issues include:

- ◇ FHE is significantly slower and non-performant (*e.g.* key-generation time, ciphertext bootstrapping): execution and key-generation time are significantly higher compared to conventional encryption schemes [304, 14, 26],
- ◇ FHE uses significantly larger key and ciphertext sizes (compared to conventional encryption schemes) [14, 171, 75],
- ◇ FHE has significantly more overhead in updating and distribution of keys (*e.g.* revocation and update of public keys) [14, 304],
- ◇ FHE has limited/restricted security features: for example, compared to authenticated encryption schemes, fully homomorphic encryption doesn't give any integrity guarantee [355].

Another key factor (from an engineering perspective) is the level of sophistication needed for the implementation of many of these schemes (for example, the HE-

[6] The represented weaknesses are not an exhaustive list covering every possible MPC scheme. We solely list some common weaknesses from synthesising recent literature. We acknowledge that there are some highly optimised protocols that are trimmed to specific use case, but we are interested in 'generic' protocols.

lib[7] [217] requires a significant effort to understand and use its low-level implementations and parameter selection).

6.2.3 Use of Expensive Operations

As already mentioned, guaranteeing privacy in the malicious setting is typically harder and more expensive than in protocols where all participants behave honestly. Hence, protecting against such attacks involves the use of complex operations and techniques (for example, zero-knowledge proofs [196] or cut-and-choose techniques[8] [286]) and results in an overhead in communication. Another example, in the context of protocols based on garbled circuits, is that operations can be transformed to utilise linear operations such as addition or logical xor; as such, operations can be performed locally and are essentially 'free' [272]. However, non-linear operations such as multiplication or logical AND require communication between the parties over a network, thus these operations are 'expensive' and undesirable.

6.2.4 Diminishing Features due to Corrupted Parties

Dependent on the number of corrupted parties, the features which a MPC protocol can satisfy diminishes. Lindell and Pinkas [290] show that for a threshold of $t < n/3$ corrupted parties, where n denotes the number of participants, secure multi-party protocols with the properties of fairness and guaranteed output delivery can be achieved without any set-up assumptions. For $t < n/2$, secure multi-party computation with the properties of fairness and guaranteed output delivery are possible assuming that parties have access to a broadcast channel. For any threshold above $t \geqslant n/2$, secure multi-party computation *without* the properties of fairness and guaranteed output delivery can be achieved assuming that parties have access to a broadcast channel and the assumption that trapdoor permutations exist. Summarising, MPC schemes lose features if more parties are corrupted — a loss which may not be acceptable in every real-world use case.

6.2.5 Ignorance of Release Privacy

Importantly, in the context of this dissertation, 'vanilla'[9] MPC schemes are only concerned with *input privacy*, but do not consider any leaks from a function's output (*i.e.* release privacy) [290]. Failure to identify such privacy leaks may result in the failure of the whole MPC protocol [80]. Hence, often there is a clear need to protect a participant's privacy from the (public) release of the collaborative function's output [357, 68]. (In specific cases, the public announcement of the result may be acceptable.)

Concretely, this problem can be illustrated by the following example. Consider the parties $P_1 \dots P_n$ that aim to compute the average of their salaries. In the case of

[7] https://github.com/homenc/HElib

[8] In a cut-and-choose protocol the proving party prepares multiple garbled circuits, while the verifying party evaluates a subset of those to verify correct construction of the circuits.

[9] Plain MPC schemes without combination of any advanced privacy techniques.

$n-1$ colluding parties and without any additional checks about the knowledge of the colluding parties, the output of the functions leaks the input of the party P_n. In the case of two-party computation, each party can easily obtain the other party's input with the knowledge of the average and their own salary.

Another example is if we consider an analyst who is able to send queries of the form Q_i to a database D, where i indicates the i-th value in the dataset; Q_i returns the output of a function applied to values $0 < i \leqslant n$ of the dataset D. For example, if D = {(Bob, smoker), (Alice, non-smoker), (Carol, non-smoker), (Eve, smoker), (Dave, smoker)}, then Q_i being the function counting all smokers within D would reveal $Q_4 = 2$. If an analyst would be able to submit queries $Q_4 = 2$ and $Q_5 = 3$, the analyst would learn that Dave is a smoker, even without seeing Dave's entry in D. Such queries violate release privacy (something that is typically not protected by a MPC scheme).

6.2.6 Discussion

Besides previously discussed performance limitations, the Usable and Efficient Secure Multiparty Computation (UaESMC) project [412] identified several real-world barriers via interviews with experts in the relevant application fields of MPC schemes:

(a) legality: current laws do not include adequate language to consider MPC use cases;

(b) receptiveness: MPC techniques are often too complex for non-experts to understand and communicated to clients without adequate auditing and monitoring tools to simplify adoption;

(c) user risk: MPC functions must be carefully drafted in order to guarantee that there is no leakage from the ignorance of release privacy;

(d) technology: scaling and data handling issues; and

(e) data visibility: hidden input information fosters suspicion of the computed results.

While some of these issues may be solvable by technological enhancements and new research outputs, especially, the concerns listed previously require a wider user engagement and the involvement of non-experts.

In the following section, we show how the techniques of trusted computing aim to solve *some* of these issues. We note, though, that even trusted computing techniques which hide some complexities of the underlying operations are prone to shortcomings with respect to the raised issues. Concretely, it is therefore often necessary to combine multiple techniques to achieve all necessary privacy guarantees. In the context of pure MPC schemes, previous mentioned shortcomings shall be understood by the reader as an opportunity of improvement and not as a 'deal-breaker'. By far, some generic (but, mainly, specific protocols and implementations) of MPC have been developed over the last decade.

In this context, Archer *et al.* [38] developed a maturity taxonomy of secure computation techniques and ranked multiple protocols according to this taxonomy as: (a) academic prototype; (b) real-world deployment; and (c) market ready. Further, in [39], Archer *et al.* present a range of applications that are enabled via MPC techniques. (At the time of writing, April 2020, three companies have been deploying market-ready MPC techniques: Sharemind[10], Unbound (former Dyadic)[11] and Partisia[12].)

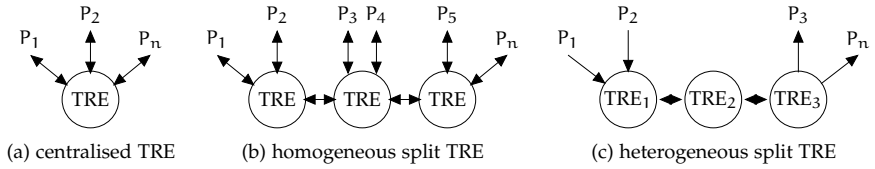
Hastings *et al.* [222] provide a thorough investigation and comparison of recent general purpose compilers for MPC. In their findings, they critisise the lack of documentation, which comes with many 'proof of concept' implementations, and advocate standardisation among the interfaces and to promote benchmarking between the frameworks.

Other, well-funded projects investigate(d) deployments of MPC techniques include: PRACTICE[13], UaESMC[14], PROCEED[15], SPEC[16] and SODA[17].

6.3 Trustworthy Remote Entity (TRE) Systems

In 2014, Paverd and colleagues introduced the concept of Trustworthy Remote Entities (TREs) in a series of papers [350, 352, 353, 351]. A TRE behaves essentially like a trusted third party (TTP). However unlike a TTP, where users are required to *blindly trust* all actions performed on the TTP, a TRE's trustworthiness can be verified by a third party. In other words, this means that a TRE is a verifiable TTP that can assure its trustworthiness through techniques provided by Trusted Computing (TC). In this context, TREs support processes of isolated execution to provide confidentiality to the data values it is processing, secure storage for processing of large-scale information which may exceed the operating memory of a remote party, and software attestation in order to verify its trustworthiness to other parties. TREs are not bound to particular architectural design choices — as such, TRE systems may appear as a single centralised unit or multiple distributed components together forming the TRE system. Importantly, to support a variety of real-world scenarios, TRE systems are capable of deploying any functionality; in our case, we focus on deploying privacy-preserving data analysis and data release operations. Consequently, in the case of a distributed TRE system, these components may deploy a homogeneous functionality (*i.e.* every component performs the same operations) or a heterogeneous functionality (*i.e.* every component performs a different operation, but operate together to achieve a global goal). What is more, TRE systems support bi-directional communication between connected parties and other TREs, contribute in the trust establishment process between a collective (*e.g.* multiple parties), and remain efficient and scalable to perform opera-

[10] https://sharemind.cyber.ee
[11] https://unboundtech.com
[12] https://partisia.com
[13] https://practice-project.eu
[14] Usable and Efficient Secure Multiparty Computation (UaESMC) http://uaesmc.cyber.ee
[15] PROgramming Computation on EncryptEd Data (PROCEED) https://www.darpa.mil/program/programming-computation-on-encrypted-data
[16] Secure, Private, Efficient Multiparty Computation (SPEC) https://cordis.europa.eu/project/id/803096
[17] Scalable Oblivious Data Analytics (SODA) https://www.soda-project.eu



(a) centralised TRE (b) homogeneous split TRE (c) heterogeneous split TRE

Figure 38: Different architectural design models of a Trustworthy Remote Entity (TRE).

tions on a large number of participants or on operations that involve data values at a large scale[18].

6.3.1 Architecture

TREs are not bound by architectural design choices and can implement any of the design models given in Figure 38. The model illustrated on the left-hand side shows a single centralised TRE unit, where each party communicates directly with the TRE (for example, each party verifies the TRE attestation assertions and shares their input with the TRE). In turn, the TRE computes its specified functionality over the input data and returns the result. This covers the model we have discussed in Chapters 2–4. The other two models, illustrated in the middle and right-hand side, show distributed TRE units. TREs underlying a homogeneous split architecture, where every component performs the same operations, and TREs underlying a heterogeneous split architecture, where every component performs a unique operation, but they operate together to achieve a global functionality are the focus point of this chapter. In these models, parties communicate either with their respective TREs (as shown in the case of homogeneous TREs) or some parties act as input provider and some parties benefit from the computed results of the TRE units (as shown in the case of heterogeneous TREs).

6.3.2 TPM-based TREs

The prototype of the TRE concept introduced by Paverd, in 2014, was based on techniques pertaining to the Trusted Computing (TC) domain such as Intel TXT[19] and Trusted Platform Modules (TPMs) [206, 207, 208]. Paverd utilised his prototype to address privacy and security problems in the smart grid domain [352, 353].

At the time of writing of this dissertation, Paverd's prototype [351] is the only TPM-based TRE we found from a review of the literature. As a consequence, the following description is mainly based on Paverd's contributions, but our overview is kept open for generic implementations of TPM-based TREs.

Paverd's TRE concept prototype is based on Intel's TXT extensions and TPM v1.2. Paverd concept implementation is trimmed to tackle the communication privacy challenges in the smart grid [352, 353], in particular, focusing on use cases of network

[18] In this context, data at a large scale refers to a classification based by Laney [277]: (a) data volume (distinguishing between (a.1.) large data values in terms of size; and (a.2.) voluminous data values in terms of quantity); (b) data velocity (in terms of how frequently input and output interfaces are queried); and (c) data variety (in terms of the structure of the data and supporting multiple data types).

[19] Intel Trusted Execution Technology [202]

monitoring, billing and demand control. In this context, the architecture of the TRE was chosen to represent a single, centralised and general purpose entity, which serves as an intermediary between the participating communication partners. These design decisions were most appropriate for the use cases in the smart grid domain — TREs (based on TPMs or on other underlying hardware security primitives) do not impose any architectural restrictions. As mentioned in the TRE architecture (Section 6.3.1), this means that a TPM-based TRE may as well interact with multiple other TREs to execute homogeneous and/or heterogeneous functionality.

TPM-based TREs leverage TPMs as the underlying hardware security primitive — as such, applications utilising such a TRE can benefit from the security features offered by the dedicated TPM.

In the context of Paverd's prototype, the executable comprises 24k lines of code and can be run on a x86 platform without a hypervisor or operating system. The single function prototype features binary attestation and a novel attestation protocol — the Finite State Attestation (FSA) protocol [351]. The prototype is entirely written in C and has been licensed under the open source BSD licence. Paverd's prototype offers formal verification and provides the privacy features of unlinkability and undetectability throughout the communication network.

The adversarial model of Paverd's prototype considers passive adversaries on the server side and active adversaries on the client side. To preserve communication privacy the following techniques are used: (a) spatial aggregation and (b) perturbation in the network monitoring use case; (c) temporal aggregation in the billing use case; and (d) a novel bidding protocol that utilises a combination of pseudonymisation and temporal aggregation in the demand control use case.

Most notably of Paverd's prototype is the implementation of the FSA protocol to prove the TRE's trustworthiness to external parties. This one-to-many attestation protocol was specifically designed to address the security and performance requirements of a TPM-based TRE. As a consequence, utilising the FSA protocol the attestation performance increases by a factor of 16 (or approximately 1500%) compared to a TPM v1.2 attestation.

The general idea behind the FSA protocol is that when the TRE finishes loading its software (during the initialisation phase) it reaches a final state in the measurement process, which it won't leave voluntarily. Using this fact, it is possible to convince multiple verifiers with the assertions of the TRE acting as a prover. In detail, during this process the TRE creates an ephemeral asymmetric key-pair when it reaches its final state. The private key then gets stored in volatile memory such that in case of a platform reset the key is irreversibly lost. The TRE generates a FSA certificate using the ephemeral public key — this certificate is used in the TLS handshake[20] with any verifier. Using a single quote with the FSA certificate, it can satisfy multiple verifiers. In order to achieve anonymity, the TRE utilises the concept of an Attestation Identity Key (AIK), an anonymous asymmetric key pair not directly linkable to a specific TPM. Moreover to prevent replay attacks, a nonce is added in the TLS protocol.

[20]The key agreement procedure of the TLS [240] protocol — https://tools.ietf.org/html/rfc8446

6.3.3 SGX-based TREs

The announcement[21] of Intel Software Guard Extensions (SGX) in 2013 enabled the establishment of a secure area within Intel CPUs, which protects applications against the compromise of other applications and even against higher privilege level software such as the operating system. Most importantly, these extensions run on the main CPU — in comparison to TPMs, which are dedicated co-processors, therefore, speed- and resource-restricted — thus, enabling resource-expensive operations to be executed with little performance impact. TREs utilising the SGX can immediately benefit from these improvements and as a result open up new application use cases, where it was previously not possible to deploy TREs due to scalability and/or performance issues.

A SGX-based TRE must satisfy the same (or, at least, equivalent) constraints and requirements as set out in Paverd's dissertation (see [351]: page 144). Fortunately, as we have illustrated previously, the features offered by the SGX satisfy those requirements. In fact, using SGX as underlying hardware security mechanism has various benefits, that would encourage practitioners[22] to prefer a SGX-based TRE over a TPM-based TRE:

1. **Performance-superiority**: SGX instructions run directly within the main CPU instead of a low-performance co-processor.

2. **Utility-superiority**: SGX-based TREs support any instructions that are available on Intel CPUs.

3. **Security-superiority**: As a user-space application via secure enclaves it can protect against a malicious operating system.

4. **Flexibility-superiority**: Functionality can be split among multiple enclaves. Those enclaves can be reused.

5. **Scalability-superiority**: Multiple enclaves may be combined according to the needs of a purpose-specific SGX-based TRE boosting scalability. In fact, any system that supports a SGX-enabled Intel CPU may be provisioned as SGX-based TRE.

6. **Special hardware-independence**: It has the potential to be more widely used as it does not rely on special cryptographic co-processors (*e.g.* TPMs).

In 2016, the current author and colleagues [34] and Küçük *et al.* [273] proposed the first prototypes of SGX-based TREs focusing on applications in the domain of privacy-preserving data analysis and data release and secure multi-party computation. While Paverd's TPM-based TRE prototype is modelled as a single intermediary party, our TRE prototype explores various architectural models as discussed previously — the TRE can comprise a single secure enclave or multiple enclaves (which can even be distributed over multiple systems). A special focus is therefore placed on the secure composability and communication between enclaves.

[21] https://software.intel.com/en-us/blogs/2013/09/26/protecting-application-secrets-with-intel-sgx
[22] *e.g.* system designers, software architects and software developers.

In order to boost efficiency and scalability of our prototype, we focused solely on SGX technology. Thus, the security assumptions of our prototype are based on the same assertions as the SGX model. Further details of the security model are discussed in Section 6.4. Confidential and integrity-preserving computation for an instance of our TRE prototype is guaranteed by the SGX mechanisms, while individuals' privacy can be assured by the utilisation of application dependent privacy models, privacy-preserving operations and algorithms.

In the context of secure multi-party applications, participants (as well as our TRE enclaves) establish a secure channel following the trust establishment through the inter/intra platform attestation process (as described previously). Via these secure channels between participants and the TRE enclaves, parties can share their inputs with the TRE. Every code and data page within an enclave is ensured to be confidentiality-, integrity- and freshness-protected by hardware enforcement [128]. The TRE enclaves process their functions on the shared inputs and return the results via the same secure channels.

Similar to standard secure multi-party schemes and the TPM-based TRE, the SGX mechanisms of a SGX-based TRE alone do not guarantee release privacy. To ensure release privacy, separate privacy algorithms must be deployed. SGX-based TREs are not bound to specific privacy mechanisms. To assess and reason about potential privacy properties of a TRE's release, we can use the privacy games introduced in Chapter 3. We note that the games, in this context, are also not bound to a specific privacy mechanism, thus allowing us to establish a privacy-preserving framework[23] that considers various privacy models and different privacy-preserving operations.

SGX-based TRE Prototype. For our TRE prototype model[24], we developed an implementation that supports various variants of personalised differential privacy. Personalised differential privacy (PDP) [165] is an adaption of the pure notion of ϵ-differential privacy [159] that allows a more fine-grained selection of the privacy budget value ϵ[25].

Our model evolved from a synthesis of the following contributions [442, 165, 322, 235, 25, 254, 357]. Concretely, our prototype utilises the Graphene libOS[26] [418] and an extended version of the GUPT library [322]. Our implementation extends the GUPT library[27] by a personalised privacy driver (see Appendix C). Since our privacy driver

[23] A SGX-based TRE is open to deploy any privacy mechanism to satisfy various privacy notions (such as anonymity, unlinkability or unobservability).

[24] Our prototype model is based on SGX v1.8 available at https://01.org/intel-softwareguard-extensions/downloads/intel-sgx-linux-1.8-release.

[25] When querying a dataset, that is protected by ϵ-DP, a *global* total ϵ budget value ϵ_{total} is reduced by a query budget ϵ_q (ϵ_q, here is independent of the dataset, but depends on the sensitivity of the query function [159]). When querying a dataset, that is protected by PDP, the budget reductions are more fine grained. For example, the query budget ϵ_q may affect only those entries in the dataset that are included in the response query (assuming the dataset is previously filtered). Another example, reduces a query budget that is defined per user.

[26] The Graphene source code can be obtained from https://github.com/oscarlab/graphene. Our SGX-prototype utilises the beta v0.2, which is available at https://github.com/oscarlab/graphene/releases/tag/v0.2.

[27] The GUPT source code can be obtained from https://github.com/prashmohan/GUPT. Our prototype uses the version available from July 2011 at https://github.com/prashmohan/GUPT/commit/ac6298c02035263772c8d7c3d0e33e8b3c05ff35.

and the GUPT library are written in python and the SGX programming model only supports the C language[28], we utilise the Graphene SGX libOS library. Graphene SGX is a wrapper that enables the execution of python within an enclave.

Our personalised differential privacy driver supports the following types of PDP (and combinations of these types):

- Type 1 — Provenance based PDP (each user has an individual privacy budget, for all of his/her shared values)
- Type 2 — Value based PDP (for each user, each shared value get assigned a privacy budget)
- Type 3 — Analyst based PDP (each data analyst get assigned an individual privacy budget)
- Type 4 — Query based PDP (each query has a specific privacy budget)

Combinations of multi-party schemes and differential privacy have been explored in the literature in the following works [80, 199, 256, 357, 168, 13, 307]. Our approach combines the benefits of SGX over multi-party schemes (*e.g.* performance, scalability, utility) with the improvements of personalised privacy budgets of differential privacy (*e.g.* flexibility and data utility).

6.4 Threat Model and Security Analysis

For system architects and system designers to make informed selection decisions and for the purpose of evaluation of a system's security and privacy guarantees our following threat model provides an overview of currently known, state-of-the-art threats to the confidentiality, privacy and integrity of TRE systems. In this section, we provide a preliminary threat model focused on release privacy. In this context, we consider a TRE system as a black box — we're solely interested in properties from the system's release. Any internal operations and processes are abstracted by the black box. In the following section, we consider threats targeting the internal operations and processes of a TRE. Further, since the release privacy mechanisms may be applied to either type of TRE system (TPM-based and SGX-based), it leaves this section generic, while in the next section we specialise on SGX-based TREs.

6.4.1 Preliminaries

Since we're treating the analysed system as a black box, we're interested in properties and relations between elements of the TRE system's released data. In this context, threats revolve around the notion of release privacy, as defined in Definition 4 in Chapter 2, and around the notions presented in Chapter 3. To treat our threat model independently, we follow the modelling of the adversary classes presented in Chapter 4. Consequently, in this section, we consider release adversaries, $\mathcal{A}_{\mathtt{release}}$, and

[28] At the time of writing, April 2019.

in/out adversaries, $\mathcal{A}_{\texttt{in/out}}$. The former class has access solely to the released data, while the latter class may also access input data to the system.

In adversarial modelling, there are typically assumptions about the strategy of an adversary being made. On the contrary, in this chapter, we aim to abstract real-world behaviour — allowing an adversary to do what she wants to achieve her goal. In this context, there exist no trusted third parties, adversaries do not follow any rules (*e.g.* semi-honest security) and incentive-based privacy models (*e.g.* [112, 320, 153, 385]) do not apply. An adversary can be seen as a malicious entity, with the only purpose to disrupt the system trying to learn any information from the system's released data.

An adversary may follow a strategy of targeting individuals as outlined by Emam [173]:

(a) **Marketer scenario.** Following this strategy, an adversary aims to target as many individuals as possible. Her goal is to de-anonymise all participants in the dataset and retrieve all sensitive information about the dataset to exploit the participants.

(b) **Prosecutor scenario.** Following this strategy, an adversary aims to target *one* specific individual. Her goal is to link this individual to a specific record in the dataset and to retrieve his sensitive attributes.

(c) **Journalist scenario.** While similar to the marketer strategy, in this strategy an adversary aims to target any individual. Her goal is to de-anonymise any participant in the dataset and retrieve any sensitive information about the dataset to exploit this individual.

6.4.2 Preliminary Threat Model (Release Privacy)

Privacy algorithms and mechanisms are designed to withstand certain classes of disclosures illustrated in the following. Further, this section outlines several classes of known attacks targeting the guarantees of release privacy of privacy-preserving algorithms and mechanisms.

Disclosure Types. The following lists the most common types of disclosure classes:

(a) **Membership Disclosure.** This class of disclosure allows an adversary to link information pertaining to an individual to a specific table or a set of data. The literature often refers to such disclosures as 'table linkage'. Examples include linking an individual, Alice, to a dataset of smokers. While such disclosure may not reveal specific sensitive values, it associates an individual with the dataset. (In fact, in real-world settings, such types of disclosures are often unwelcome.)

(b) **Attribute Disclosure.** This class of disclosure allows an adversary to disclose sensitive values within a released dataset. Examples include the disclosure of specific types of cancer within patients records. While such disclosures may not reveal the identity of individuals (*e.g.* a patient's record), it reveals sensitive

information of the dataset itself. (Given the context and aim of an adversary, such disclosures are undesirable.)

(c) **Identity Disclosure.** This class of disclosure allows an adversary to map an individual to a specific tuple within a dataset. The literature refers to such disclosures often also as 'record linkage'. Depending on the privacy mechanism / algorithm, this type of disclosure is typically most devastating, since it can completely break the guarantees that the privacy mechanism / algorithm aims to achieve. (For example, in a k-anonymous, {generalisation or suppression}-based privacy release an adversary would 'only' learn limited information and in a {anatomisation, permutation}-based privacy release an adversary may learn a specific value with high confidence. See Appendix A for examples.)

Known Attacks. The following lists a subset[29] of currently known attacks on release-privacy systems:

(a) **Homogeneity Attacks [298].** Formalised by Machanavajjhala *et al.* [298] in 2007, homogeneity is satisfied if all sensitive values within a set of anonymous released quasi-identifiers are equal. This means that an adversary who is able to link an individual to these records is also able to retrieve the sensitive value with certainty. Importantly, the privacy guarantees of the privacy mechanism are still satisfied. (This kind of an attack is not a weakness of the privacy mechanism, but in the design of the mechanism.) For example, k-anonymity is vulnerable to this kind of attack — suffering from a potential lack of diversity of the sensitive values within a k-anonymous set.

(b) **Background Knowledge Attacks [298].** Similarly Machanavajjhala *et al.* [298] formulated so-called *background knowledge* attacks. The attack assumes that the adversary already knows that a certain individual's data is included in the dataset. Then, by leveraging population information[30], the set of possible sensitive values that are associated to the quasi-identifying values can be significantly reduced. In other words, by introducing additional knowledge about the individual in the dataset, the record matching can be significantly reduced. (Many release privacy mechanisms are vulnerable to background knowledge attacks, since these algorithms focus on properties of the release such as hiding relationships between output elements. Probabilistic algorithms such as differential privacy [159] are not susceptible to this kind of attack, since differential privacy is a property of the release algorithm.)

(c) **Similarity Attacks [282].** Conceptually similar to the homogeneity attacks, Li *et al.* [282], also in 2007, formalised so called similarity attacks. ℓ-diverse releases require a diversity of the sensitive attributes, however, these values may still be semantically similar. In other words, the sensitive attributes may pertain to the same class or category of values. In this context, an adversary would be able

[29] We surveyed the literature to provide the reader with a brief overview of state-of-the-art attacks. It is not out aim to identify 'every' single class of attack.

[30] Population information are properties that hold true for a subset of the population. For example, 89 Austrians studied in 2017 at the University of Oxford [6].

to retain useful information about the released dataset. For example, from a set of possible outcome elements {lung cancer, breast cancer, pancreatic cancer}, an adversary may conclude that the observed individual has cancer, however, they may not learn the specific type of cancer.

(d) **Skewness Attacks [282].** In addition, Li *et al.* [282] introduced so-called skewness attacks. This attack leverages skewed distribution differences in the equivalence classes[31] of the anonymised release. For example, consider that within a given dataset, 1% of the records indicate a susceptibility to a specific disease. In this context, we assume the dataset contains positive and negative records. The following combinations of equivalence classes (of size 100) all satisfy 2-diversity: equivalence class (a), which contains an equal number of positive and negative records; equivalence class (b), which contains 99 positive and 1 negative record; and equivalence class (c), which contains 99 negative and 1 positive record. If an adversary is able to associate an individual to any of these groups she can determine stark differences in the privacy level (50% for class a, 99% for class b and 1% for class c). Thus, a skewed distribution of the equivalence classes with regards to the overall table (and, henceforth, the abstracted population) may lead to significant privacy risks.

(e) **Composition Attacks [188].** In 2008, Ganta *et al.* [188] presented so-called composition attacks (in [188], Ganta *et al.* specialise on intersection attacks, a subset of composition attacks). Composition attacks study the effect of independently released anonymised datasets. In other words, an adversary leverages the knowledge gained from one anonymised release to de-anonymise the other release. Partition-based anonymisation schemes such as k-anonymity [408], ℓ-diversity [298] and t-closeness [282] are in particular susceptible to these kind of attacks. Composition attacks rely on two properties: (a) exact sensitive value disclosure, which means that sensitive values are in no way altered (*e.g.* perturbed); and (b) locatability, which means that given an individual's non-sensitive values one can locate the equivalence class. Fortunately, differential privacy [159] is again not susceptible to these attacks.

(f) **Minimality Attacks [439].** Wong *et al.* [439] introduced so-called minimality attacks, which leverage the minimality principle of data distortion when anonymising sensitive information to uphold data utility. In principle, privacy-preserving mechanisms alter the quasi-identifier of a tuple in case these values infringe the 'structure' (constraints) of the privacy model. Minimality attacks require an adversary to obtain knowledge of the utilised privacy model (*e.g.* k-anonymity), privacy mechanism (*e.g.* generalisation) and access to external tables that help the adversary to reduce the number of possible matches to the sensitive records. Utilising this knowledge, solely by reasoning, the adversary is able to rule out many records that can not match (or reduce the number where privacy-preserving operations were applied). Consider the following example. To satisfy 2-diversity, the equivalence class must contain at least two distinct sensitive attributes. If

[31] An equivalence class is, for example, a subset of k-indistinguishable records with $k < n$ and n being the number of all released elements [282].

the equivalence class size is large enough (for example, if there is a restriction on the number of sensitive values and assuming non-sensitive values are not anonymised) this class can be ruled out. Utilising such techniques, an adversary is able to significantly reduce the number of possible matches. For more indicative examples of minimality attacks, we refer to [439].

Many privacy attacks (for example, those listed above) focus on the model of non-interactive privacy, due to its release-and-forget approach (*i.e.* 're-sanitisation' of a released (sanitised) version of a database in case of a privacy breach is not possible) and partition-based privacy mechanisms. Other attacks targeting release privacy include freeform attacks [430] by Wang *et al.*, the deFinetti attack [262] by Kifer and walk-based, cut-based and passive attacks on anonymised social networks [45] by Backstorm *et al*. Further, Liu *et al.* [293] survey attacks against privacy schemes based on privacy-preserving perturbation methods.

6.4.3 Mitigations

While most of the previously described attacks are focused on partition-based privacy models, privacy algorithms pertaining to this class may still release appropriate 'good' releases. By taking the previous vulnerabilities into consideration in the design process of a privacy-preserving system and by careful analysis of possible background information, an informed decision can be made to release de-anonymised information about a dataset. (These decisions are typically specific to a use case and the available data.)

An alternative solution is to focus on privacy-release algorithms that are part of the release mechanism such as variants of differential privacy [159] (see Section 2.8.4). While these schemes are not perfect[32], they resist a lot of the previous mentioned attacks, solely because those schemes preserve privacy via a transformation of the data dependent on the query function. In this context, the privacy mechanism is provably independent of the data (including arbitrary background information) [159].

6.5 Advanced Threat Model (for TRE Systems)

The previous section focused on a black box model and was concerned about the information an adversary may infer from the outputs of such a system. In this section, we look at the internal operations and algorithms. What is more, the focus is on input privacy, integrity of computation, confidentiality and freshness. Our analysis is specifically trimmed to SGX-based TRE systems, but may also apply to related privacy-preserving systems.

[32]The following link provides a brief overview of vulnerabilities: https://en.wikipedia.org/wiki/Implementations_of_differentially_private_analyses.

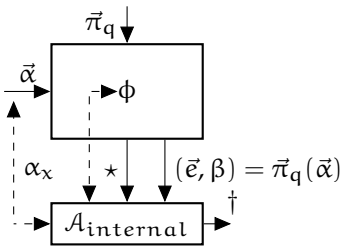


Figure 39: An adversarial model of type: internal adversary. This kind of adversary obtains access to the inputs $\vec{\alpha}$, the system release $\vec{e}, \beta$ and any internal variables of system ϕ.

6.5.1 Preliminaries

In the context of input privacy we consider two types of adversary models: a) in-/out adversaries, $\mathcal{A}_{in/out}$ and b) internal adversaries, $\mathcal{A}_{internal}$. The former class is similar to those introduced in the previous section, while the latter class has access to *some* internal operations of the black box system, which she exploits to learn/predict information about the input and output values. Figure 39 illustrates this class of adversary.

In this regard, with protecting input privacy we associate the principles of confidentiality, integrity and freshness. To reiterate definitions of these terms, in the context of this dissertation, by *confidentiality* we mean that only authorised parties should be able to access the clear text information; by *integrity* we mean that unauthorised alterations can be detected; and by *freshness* we mean that replays of messages are prevented.

Further, in this section, we focus on implementation attacks. Such attacks target concrete software implementations of an algorithm. In this context, an adversary may mount an attack on different layers of a system. A user-space attack has the highest privilege abstraction (ring 3), while a kernel-space attack operates on ring 0 — exploiting access to other user-space applications and kernel memory. In SGX-based TRE systems, input privacy is guaranteed via the underlying hardware security primitives which protect user-space enclaves from access via kernel-space.

6.5.2 Extended Threat Model

In our extended threat model, we allow an adversary to arbitrarily attack a SGX-based TRE system. There are no limitations / assumptions restricting the adversary. In the following, we iterate from high-level guarantees / attacks to the algorithms which provide these guarantees.

Input Privacy. Potential attacks violating input privacy can be categorised as follows:

(a) **Confidentiality Attacks.** In a confidentiality attack, an adversary aims to exceed her privilege level to access messages/data not intended for her to read. For example, if an adversary is able to instruct a malicious user-space application that gains kernel-level privileges she can thereafter read data pages of

other user-space applications. Recently, the privilege-escalation attacks such as Meltdown [291] and Spectre [271] attacks have received a lot of media attention, which enable an adversary to read kernel-space memory mounted from user-space applications. Among others, Chen *et al.* [106] extend these kind of attacks, dubbed SgxPectre attacks, to read secrets even from SGX-protected enclaves.

(b) **Integrity Attacks.** In an integrity attack, an adversary aims to perform illicit writes to memory locations to alter, for example, the execution flow (to circumvent or change branch conditions), report wrong values or solely to make the data unusable (*e.g.* to mount a denial of service attack). Typically, an adversary aims to trigger such attacks silently, without the imminent detection of a defense system. Examples of such attacks include buffer overflows [261, 268], format string attacks [393] and, in a wider sense, integer overflow attacks [356].

(c) **Freshness Attacks.** In a freshness attack, an adversary aims to replay parts of a message or the whole previous message. Such vulnerabilities arise if the system is not keeping a state between the communication partners (typical in asynchronous messaging) and if the incoming messages are not authenticated with regards to who sent it. Freshness attacks interplay with integrity attacks, since in a typical man-in-the-middle attack an adversary would capture the legitimate messages of a communication partner, change those messages according to her needs and replay it to the privacy-preserving system. Examples of such attacks include so-called rollback attacks [346, 406, 306], which replay an old state of a system.

Given a description of attack vectors, we dive deeper into the security of the underlying hardware primitives of a SGX-based TRE.

Security Review of Intel SGX. An complete security review of SGX is beyond the scope of this dissertation. However, the following review briefly outlines the algorithms used in SGX to guarantee confidentiality, integrity and freshness. For more details, we refer to [44, 128].

Figure 40 shows the difference between a traditional attack surface including all higher privilege level code such as the operating system and virtualisation. SGX significantly reduces the attack surface — it is only required to trust a limited set of microcode instruction (hardware-level) and the code and data pages within an enclave. Given an Intel CPU, any code and data page outside the immediate CPU package is protected by cryptographic primitives that guarantee privacy, integrity and freshness.

A user-space application is protected via the segregation of virtual memory into layers (*e.g.* user-space, kernel-space), 'sand-boxing' of processes and higher level access control mechanisms to grant and restrict access to the application. SGX enclaves are part of processes within the user-space, or 'ring 3', and restrict access even from higher privilege level software such as firmware and device drivers on 'ring 2/ring 1', the operating system on 'ring 0', any hypervisor on 'ring -1', the System Management Mode (SMM) on 'ring -2' or even Intel's Management Engine on 'ring -3'.

SGX uses various kinds of cryptographic components and algorithms in order to ensure its privacy guarantees. To generate keys and random numbers, strong random-

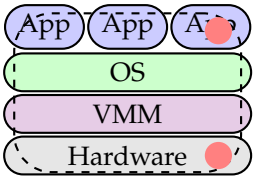


Figure 40: The traditional attack surface of a user-space application, indicated by the dashed line, includes the hardware, virtualisation, operating system and user-space applications. The attack surface of SGX, indicated by ●, is significantly reduced to the enclave (within the user-space process) and a limited set of hardware packages.

ness is required — SGX offers a function sgx_read_rand which further invokes the hardware-based pseudorandom generator (PRNG) available in all Intel CPUs. In addition, SGX makes use of various kinds of standardised cryprographic algorithms for operations such as enclave signatures and attestation processes (RSA and SHA-256); enclave policy checks (ECDSA); key exchange (ECDH); data sealing (AES-GCM); key derivation (AES-CMAC); and memory encryption (AES-CTR) [44, 128].

It is important to note that for memory encryption and quote creation purposes, Intel deploys custom developed closed-source cryptographic primitives. In order to achieve confidentiality and integrity of code and data pages evicted to the RAM, the Memory Encryption Engine (MEE) uses a dedicated Carter-Wegman Message Authentication Code with a tweaked AES-CTR [214]. To create a quote for attestation, Intel deploys a combination of RSA, AES-GCM, SHA-256, and the signature scheme EPID [84].

While the original threat model of Intel SGX considers that any higher-privilege level code such as the operating system is untrusted and no hardware apart from the CPU is included in the TCB, Intel's whitepapers [29, 232, 310, 127, 125, 122] discard cache-timing attacks as unpractical; further, in [123] it is mentioned that "Intel SGX architecture does nothing to improve this position." In a similar fashion, Aumasson and Merino [44] and Costan and Srinivas [128] note in their analysis that SGX excludes micro-architectural attacks (*i.e.* cache-timing attacks, physical attacks, microcode attacks) from their threat model. Further, denial-of-service attacks are ruled out of scope.

Nevertheless, SGX has received a lot of attention from academic researchers analysing the security claims of SGX. The following paragraphs summarise these results mainly around side-channel analysis of SGX.

Attacks on Intel Instruction Sets. Recently, Intel CPUs have gained a lot of media attention due to the variety of devastating attacks targeting mainly the Intel platform. Since a SGX-based TRE relies on Intel CPUs to operate, we refer to some recent attacks that must be considered by a system designer.

Meltdown. Meltdown [291] breaks the isolation between user-space and kernel-space and allows a rogue process to read from any memory location including kernel memory and memory allocated to other processes. To achieve this, Meltdown exploits a race condition and mounts a cache side-channel attack.

Spectre. Spectre [271] breaks the isolation between different user-space processes and allows a rogue process to potentially read sensitive data of the other processes. To achieve this, Spectre exploits observable side-effects that result from a branch misprediction in the speculative execution of modern processors and mounting a timing attack. Further, Chen *et al.* [106] extend Spectre-like attacks to target SGX.

Foreshadow. Foreshadow [87, 437] breaks the memory protection of SGX enclaves and the virtual memory abstraction of modern processors. Foreshadow [87] allows a rogue process to read memory within SGX enclaves and an additional attack, dubbed Foreshadow Next Generation [437], allows a rogue process to read Virtual Machines (VMs), hypervisors (VMM), operating system (OS) kernel memory, and System Management Mode (SMM) memory. Foreshadow belongs to the family of speculative execution attacks and exploits transient out-of-order execution.

Zombieland. Zombieland [390] breaks the isolation between different user-space processes, user-space and kernel-space as well as virtual memory isolation and allows a rogue process to read sensitive data of other processes, kernel-space secrets and applications executing in the cloud. To achieve this, Zombieland exploits the fill buffers in modern processors. Concretely, Zombieland gets hold of secrets currently processed by other processes. Zombieland belongs to the family of Meltdown-like attacks.

Side Channel Constraints of SGX. Side-channel analysis targets the implementation of privacy-preserving systems. In the context of Intel SGX, in the recent past, there has been excessive analysis with regards to such attacks. We focus on software-based side-channel attacks — mainly, because such attacks can be (simply) mounted by a remote attacker. We limit our analysis in this context and omit physical attacks (including denial-of-service attacks). Our reasoning for these restrictions are as follows: a) for a malicious cloud provider it would be trivial to mount such attacks; b) such attacks / analysis typically involve the physical dismantlement of the CPU (or parts of it) requiring physical access to the computational devices; c) such attacks are typically quite cost-intensive (compared with the gain of an adversary); d) TREs predominantly operate in a distributed fashion — an adversary would need to gain access to multiple physical locations to mount a successful attack.

Costan and Devadas [128] give an excellent overview of various physical attacks targeting modern computer systems — including security guarantees offered by the SGX model. Examples of these attacks include: bus tapping attacks, debug port attacks and DRAM (*e.g.* Rowhammer [267], RAMBleed [276]) attacks. (Yet, not all of the attacks listed in [128] apply directly to SGX-based TREs. Still, since those attacks may be used to breach the underlying security of modern computer system, on top of which a TRE is based upon, system designers must take those into consideration[33].)

[33]The SGX threat model explicitly states that it can not trust any outside call (OCALL). Consequently, if an adversary is able to breach higher privilege level code such as the operating system, she is in a position to influence / inject malicious content into OCALLs.

Examples of currently widely-known side-channel attacks include the following (an in-depth overview of such attacks is out-of-scope of this dissertation, thus we refer to the appropriate literature including [189, 243, 425, 212, 211]):

(a) **Page-Fault Attacks.** Page-fault attacks exploit a program's memory access behaviour (in the RAM) via its page fault patterns. Introduced in 2015 by Xu *et al.* [445] and further refined and applied to SGX by Shinde *et al.* [397], these attacks allow a malicious process overtaking an untrusted operating system to extract large amounts of sensitive information from protected applications. Page fault pattern attacks assume that branching conditions are executed on different code pages, which can be detected by the adversary. Examples of these attacks extract sensitive information from non-cryptographic functions [445] and cryptographic functions [397].

(b) **Cache Side-Channel Attacks.** Cache side-channel attacks exploit the memory access behaviour (in the cache) of modern computers. Typically, level 1 (L1) caches are shared only between threads on the same processor core, but level 3 (L3) or last level caches are shared between all CPU cores — exposing a potential vulnerable surface especially in distributed cloud settings. Devastating examples of cache side-channel attacks include Prime+Probe [449], Scatter-Gather [411], Flush+Reload [448] and Flush+Flush [213]. In a nutshell, cache attacks operate as follows:

(1) The adversarial process fills/flushes the cache with specific code/data pages.

(2) It waits for some time, while the victim process executes and fills the cache.

(3) Thereafter, the adversarial process tries to access its code/data pages and measures the time for a response. If the access is fast (cache-hit) then the victim process also used this instruction, otherwise (cache-miss) the cache line got evicted.

Some of the previously introduced attacks require that the adversarial process and victim process share code or data pages between each other. Page sharing is typical and beneficial for the purpose of inter-process communication and to reduce the memory footprint of an application. Yet, since SGX purposely doesn't allow sharing of pages between different SGX enclaves, some of the previously introduced attacks may not be directly applicable. However, Schwarz *et al.* [392] show a Prime+Probe attack on a co-located SGX enclave. Götzfried *et al.* presents a cache-timing attack, Brasser *et al.* [79] show practical cache-based attacks targeting SGX and Moghimi [321] show key recovery attacks in SGX enclaves.

(c) **Iago Attacks.** Iago attacks exploit the system call API between kernel and application space. Iago attacks were first introduced in 2013 by Checkoway and Shacham [105]. In an Iago attack, a malicious kernel can produce a carefully chosen sequence of integer return values of a protected process (*e.g.* containing an SGX enclave) to manipulate the process to act against its interest and even to undertake arbitrary computation on the malicious kernel. Since SGX protects

only parts of the application's program, but does not trust any outside interface (OCALL) these attacks can compromise the privacy guarantees given by SGX. (SGX prevents the read, write and remaps of memory inside an enclave, but still relies on certain external libraries which must be carefully written in order to be secure against potential Iago attacks.)

(d) **Synchronisation Attacks.** Synchronisation attacks exploit synronisation bugs within enclave code. These kind of attacks were first introduced in 2016 by Weichbrodt *et al.* [435]. In a synchronisation attack an adversary exploits use-after-free and time-of-check-to-time-of-use bugs in enclave code to hijack an enclaves control flow or in order bypass access control mechanisms. For the attacks to work, the application must be multi-threaded (which is typical in any modern application). As the attacks have shown, the impact of synchronisation bugs is much more significant in SGX enclaves than in traditional applications. This is caused due to its highly reliable exploitation through adversary controlled scheduling.

(e) **Side-Channel Attacks on Differential Privacy Variants.** Other types of side-channel analysis on SGX-based TREs involve attacks against their release privacy mechanisms. For example, Haeberlen *et al.* [215] show side-channel vulnerabilities in differential privacy systems like PINQ [312] and Airavat [379] such as time, state and privacy budget side-attacks. The attacks break down queries into micro-queries and macro-queries. The former defines the mapping operation of a single row of the database and the latter defines the rest of the query handling (*e.g.* combining of the results of the micro-queries). Then, in a timing attack it is assumed that the adversary is able to change the code such that for a target condition (*e.g.* a sensitive record) the execution is timed-out for a specific time, which the adversary is able to detect. In a state attack, the adversary uses a 'state' to pass to the following micro-queries after reaching a certain condition (*e.g.* for all following micro-queries return 1 after a sensitive record is found). In a similar fashion, an adversary running an privacy-budget attack may issue a privacy-budget intensive sub-query when a micro-query detects a sensitive record. While all these attacks violate differential privacy, other differential privacy systems such as Fuzz [215] and GUPT [322] claim to be secure against these kind of side-channel attacks.

(f) **Other Side-Channel Type Attacks.** Other possible side-channel attacks targeting SGX include: rollback attacks [406, 306], return-oriented programming attacks [391], interface-based side-channel attacks [429], branch shadowing attacks [281] and interrupt latency attacks [225].

6.5.3 Potential Mitigations

After having reviewed several vulnerabilities targeting SGX-based TREs, we present mitigations to these attacks. SGX applies some generic techniques to mitigate confidentiality, integrity or freshness leakage:

(a) **Confidentiality Attacks.** To prevent against attacks that violate the confidentiality guarantee SGX utilises encryption of pages outside of the immediate CPU package.

(b) **Integrity Attacks.** To prevent against attacks that violate the integrity guarantee SGX utilises message authentication codes and signatures.

(c) **Freshness Attacks.** To prevent against attacks that violate the freshness guarantee SGX utilises nonces and integrity mechanisms.

A TRE relies on the guarantees of the underlying hardware security primitive (in our case the guarantees of SGX). Costan and Devadas [128] and Aumasson and Merino [44] summarise the cryptographic primitives underlying SGX (providing the guarantees of confidentiality, integrity and freshness).

Other techniques to prevent an adversary from exploiting practical attacks (such as a side-channel attack) include Oblivious RAM [194] and Path ORAM [405] to prevent memory pattern attacks, input/output value validation, redundant re-computations, range checks to prevent Iago-type attacks, time-independent uniform computations to tackle timing attacks, memory-uniform accesses and disabling of hyper-threading (to prevent sharing of resources), random flushing of caches to prevent against cache side-channel attacks, and computing branch condition on the same code page to prevent controlled channel attacks.

Further, the literature proposes mitigations as follows (focusing on the above introduced attack vectors): Tramèr *et al.* [413] present so-called 'sealed glass proofs', which models transparent (confidentiality leaking) yet integrity-preserving enclave computations secure against side-channel attacks. Völp *et al.* [426] present techniques to avoid leakage and synchronisation attacks via enclave-side preemption control. Brandenburger *et al.* [78] present techniques to detect forking and rollback attacks. Fu *et al.* [183] present mitigations of controlled side-channel attacks via enclave verifiable page faults. Jang *et al.* [248] show mitigations against the Rowhammer attack in SGX. Kuvaiskii *et al.* [275] protect enclave memory to protect against attacks like Heartbleed[34]. Finally, Oleksii *et al.* [339] present techniques to protect against practical cache timing and page table side-channel attacks.

[34] http://heartbleed.com/

6.6 Performance Evaluation of TRE Systems vs. MPC

In part, this dissertation is driven to address efficient and scalable privacy-preserving data analysis and data release[35]. In large-scale and distributed contexts, the support of many participants (or many operations) requiring privacy-preserving operations has evolved from theoretical endeavours to practical solutions as we have seen in the previous sections.

We report on the performance of SGX-based TREs and compare our results to a selected set of popular two-party and multi-party frameworks. We have selected the two party protocols, Fairplay [301], Tasty [226] and MightBeEvil [236], and the multi-party protocols, FairplayMP [56], Sharemind [67], VIFF [135], SEPIA [89], SPDZ [136] and SCAPI [169], for our evaluation. (Table 3 presents an overview of these protocols.) Choi and Butler [114] present an excellent overview of recent research developments for multi-party schemes and trusted hardware.

Several of the discussed protocols present preliminary experimental performance evaluations. The reported results evaluate the number of gates used in garbled circuits and execution times of specific subtasks of the protocol (*e.g.* initialisation, circuit garbling, network communication and function execution). Other related work presenting performance evaluations include [137, 70, 69, 360, 227]. Further, Paverd [351] presents evaluations and comparisons of attestation protocols to his proposed FSA protocol (utilised in TPM-based TREs). In a separate performance evaluation [273], we present benchmarking results of basic SGX operations (showing that a single SGX-based TRE may support approximately 20,000 relying parties within a 30 minutes interval).

More recently, Bahmani *et al.* [47] present evaluations of SGX in comparison with MPC schemes utilising the ABY framework[36] [143]. Pattuk *et al.* [348] presents the CheapSMC framework, which empirically evaluates basic Arithmetic, Boolean, and Yao [447] sharing operations on Amazon EC2 virtual machines[37]. Hastings *et al.* [222] present a thorough investigation and comparison of recent general purpose compilers for secure multi-party computation. Similarly to our results their investigation is aimed at non-experts and advocates improved documentation and standardisation of MPC schemes. Nevertheless, to the best of our knowledge, other than these preliminary results, there has not been a thorough analysis of generic MPC protocols (apart from Hastings *et al.* [222]) in comparison to trusted hardware based solutions (such as our SGX-based TRE). In the following sections, we provide an overview of the selected protocols and present our benchmarking results.

6.6.1 Secure Two-Party Protocols

Secure two-party protocols involve, as the name already reveals, two parties that want to securely compute a common function without revealing their private inputs to the

[35] Addressed by research question 4 in Section 1.2.
[36] https://github.com/encryptogroup/ABY
[37] https://aws.amazon.com/ec2/

respective other. The following generic protocols have evolved over the past years. (A summary of the protocols is presented in Table 3.)

Fairplay. Fairplay was introduced in 2004 by Malkhi *et al.* [301]. It is one of the first generic automated framework for secure function evaluation. The underlying secure evaluation arises from Boolean circuits and the protocols utilise oblivious transfer [365] to secure the communication between two parties. Fairplay enables generic function evaluation. In this context, a collaborative function is defined and written in a high-level procedural definition language called secure function definition language (SFDL). Using the Fairplay compiler such a function is transferred into a one-pass Boolean circuit structure called secure hardware definition language (SHDL), which, in turn, can be evaluated using so-called {Bob, Alice} programs. Fairplay is actively secure by using the cut-and-choose method by Lindell and Pinkas [286]. Preliminary performance results are presented by the authors, who analyse their framework in local area/wide area network settings using functions for logical AND, the Billionaires problem[38], a median function and a keyed database search function.

Tasty. Tasty was introduced in 2010 by Henecka *et al.* [226]. It succeeds the Fairplay framework and allows software developers to generate protocols based on Boolean and Arithmetic garbled circuits, homomorphic encryption, and combinations of the previous. The framework comprises: (a) a compiler for a domain-specific language called Tastyl; (b) client and server programs for secure function evaluation; and (c) tools for testing, benchmarking and the post-processing of protocol implementations. Importantly, Tasty also supports the evaluation of circuits (in SHDL format) generated by the Fairplay compiler. Tasty has four predefined security levels (80, 96, 112, 128 bit) supporting passive security based on elliptic curves. Preliminary performance results by the authors include statistical information with regards to the setup time and on-line time for basic set intersections and face recognition use cases. Further, performance results on multiplication circuits are compared with Fairplay.

MightBeEvil. MightBeEvil was introduced in 2011 by Huang *et al.* [236]. It was designed to show that general purpose garbled circuits are able to outperform special purpose protocols for specific functions. MightBeEvil is a flexible framework based on Boolean and Arithmetic garbled circuits. The framework was designed to improve on the efficiency and scalability of Fairplay. As such, it utilises topologically sorting of garbled circuits, pipelining and several other optimisation techniques such as the "free XOR" technique [272], garbled-row reduction [360], and oblivious-transfer extension [244]. MightBeEvil is secure under the semi-honest security model. Preliminary performance results by the authors show that for four specific use cases (Hamming distance [343], Levenshtein distance [252], Smith-Waterman algorithm [372] and an implementation of AES [360]) it outperforms its competitors (SCiFI [343], Tasty [226] and an implementation by Jha *et al.* [252]). The authors report the on-line time and overall time.

[38] The 'Billionaires' problem is an extension of the Millionaires problem by Yao [447], whereby the numbers are 32 bit inputs.

6.6.2 Secure Multi-Party Protocols

Simultaneously, while two-party protocols received attention for improvements and enhancements, protocols which address multiple parties emerged. Generic protocols arising from this strand include the following. (A summary of the protocols is presented in Table 3.)

FairplayMP. FairplayMP was introduced in 2008 by Ben-David *et al.* [56]. FairplayMP succeeds the Fairplay framework by introducing a multi-party version and several other optimisations to the protocol. FairplayMP is based on the BMR protocol [51] utilising Boolean garbled circuits (importantly in the multi-party setting, BMR also supports a constant number of communication rounds) and the BGW protocol [57] for the purpose of constructing gate tables. The framework of FairplayMP comprises: (a) a compiler that supports the SFDL 2.0 language — an extension of the SFDL of Fairplay; and (b) a cryptographic engine to evaluate secure multi-party protocols. Protocols that are designed in the FairplayMP framework follow the design pattern to distinguish parties into input, computational and result players (a party may assume one, multiple or all of these roles). FairplayMP protocols are secure against semi-honest adversaries. The authors present preliminary benchmarking results for a second price auction use case and a voting use case.

Sharemind. Sharemind was also introduced in 2008 by Bogdanov *et al.* [67]. Sharemind is a privacy-preserving data processing framework based on novel additive secret sharing techniques. One of the design goals of the Sharemind framework is to enable the design of efficient protocols using only basic mathematical operations. Using these basic operations, more complex protocols can be designed. Hence, in this context, all basic operations are universally composable [93]. The Sharemind suite includes a software development kit (SDK) to design secure protocols. Sharemind relies on the assumption of three non-colluding independent hosts. Consequently, Sharemind protocols achieve security in the semi-honest model. The authors of Sharemind present benchmarking evaluation compared to Fairplay, SCET [71] and an implementation by Yang *et al.* [446].

VIFF. The Virtual Ideal Functionality Framework (VIFF) was introduced in 2009 by Damgård *et al.* [135]. It is a software suite trimmed to design secure generic applications in the multi-party setting. VIFF enables the design and deployment of asynchronous protocols and is based on Arithmetic garbled circuits and Shamir's secret sharing technique [395]. Importantly for the multi-party setting, VIFF allows for the automatic parallelisation of primitive operations without resorting to complex multi-threading. VIFF's generated asynchronous protocols are categorised into three phases: pre-computation, input phase and computation phase. In the pre-computation phase various variables may be initialised; in the input phase, so called multiplication triples [53, 131] are generated from the inputs, while in the computation phase these multiplication triples are evaluated. Protocols that are generated by the VIFF suite achieve perfect security against adaptive malicious adversaries under the assumption

that less than $n/3$ parties are corrupted. The authors present basic benchmarking results for the use case of multiplication of random bits.

SEPIA. The SEPIA library was introduced in 2010 by Burkhart *et al.* [89]. The library, which was designed for secure applications in the multi-party setting, was rolled out and evaluated mainly for multi-domain network security and monitoring applications. SEPIA is based on Shamir's secret sharing technique. Protocols designed with SEPIA are optimised for parallel invocations. SEPIA includes a complete set of basic operations that enable private addition and private multiplication of secret shares. Importantly, the library implements optimised comparison operators for the generated shares (such as less-than, equality and short range checks). These operations are all universally composable and protocols designed utilising the SEPIA library are round constant (for a fixed field size). SEPIA distinguishes between input and privacy peers — input peers deal with the inputs of the protocol, while privacy peers perform the actual computations (in this context, privacy peers simulate a trusted third party). SEPIA protocols achieve passive security. The authors of SEPIA investigate the practical usefulness of multi-party schemes for network security and compare their preliminary benchmark results with the libraries of VIFF and FairplayMP.

SPDZ. The SPDZ protocol was introduced in 2012 by Damgård *et al.* [136]. SPDZ is based on somewhat homomorphic encryption [77] to design protocols for arithmetic secure multi-party computation. In addition, to further enhance privacy, SPDZ makes use of adapted zero-knowledge proofs. SPDZ protocol phases can be distinguished between an offline preprocessing phase and an on-line computation phase. In the pre-computation phase all expensive public-key operations are handled, while the online phase operates only on 'cheap' primitives. To optimise, SPDZ deploys the circuit randomisation technique of Beaver [52]. Protocols designed by the SPDZ protocol are universally composable. SPDZ also achieves active security under the assumption of $n-1$ corrupted parties (out of n parties). The SPDZ protocol was succeeded by the SPDZ 2 protocol by Damgård *et al.* [141]. SPDZ 2 improves on the basic SPDZ protocol via various efficiency innovations on both the theoretical and practical side (for example, via the establishment and distribution of a BGV [76] public key for homomorphic encryption). The authors of SPDZ 2 present preliminary benchmarks and compare the protocol to the original SPDZ protocol [136] and an enhanced version of the SPDZ protocol [140], which provides security in the presence of a dishonest majority of active adversaries. Our benchmarking is based on an implementation of the SPDZ protocol by Enes *et al.* [174].

SCAPI. The Secure Computation Application Programming Interface (SCAPI) open-source library was introduced in 2012 by Ejgenberg *et al.* [169]. SCAPI is a generic cryptographic library implementing various cryptographic primitives, which are tailored for secure computation. The aim of the authors of SCAPI was to provide a platform of cryptographic primitives for secure applications and protocols in the multi-party setting. In this context, SCAPI was designed to support the principles of flexibility, extensibility, efficiency, and ease of use (for software developers utilising

Table 15: Comparison of Protocols for Software Attestation.

Protocol	Type	Privacy-Preservation	Reference
TPM 1.1b Att.	one-to-one	privacy CA	[204]
DAA	one-to-one	zero-knowledge	[81]
Intel SGX Att.	one-to-one	EPID	[253, 29, 84]
FSA	one-to-many	AIK	[351]

the library). The library's implementations are divided into three layers: (a) a layer for basic primitives (*e.g.* hash functions, pseudorandom functions, pseudorandom permutations, pseudorandom generates and key derivation functions), (b) a layer for non-interactive protocols (*e.g.* message authentication codes, symmetric and asymmetric encryption schemes and digital signature schemes), and (c) a layer for interactive protocols (*e.g.* commitments, garbled circuits, oblivious transfer, and zero knowledge proofs). The security levels of the protocols developed with SCAPI depend in the purpose of the applications that are developed. The authors do not provide performance benchmarks on their own, but SCAPI is included in the MATRIX[39] framework.

6.6.3 Software Attestation Protocols

Software attestation protocols are the backbone of applications and systems deploying trusted hardware primitives. In the past, several software attestation protocols have been proposed. In the following, we give a brief overview of a selection of these protocols — protocols that are of interest for TRE systems. These protocols include: the TPM v1.1b [204] protocol, the widely deployed DAA [81] protocol, the attestation protocol of SGX [253, 29, 84], and the protocol designed by Paverd for his prototype version of a TPM-based TRE, FSA [351]. The first three protocols can be categorised into one-to-one attestation protocols, while the FSA protocol falls into the category of one-to-many protocols. A summary of the attestation protocols is presented in Table 15.

TPM v1.1b Attestation. Included in the specification of the TPM v1.1 [204], the TCG introduced in 2003 its software attestation protocol. The TPM v1.1 attestation protocol requires a trusted third party in form of a privacy certification authority (privacy CA). The privacy CA is used to ensure privacy-preserving attestation processes of a TPM to the outside world. For the attestation process, a TPM is associated with a unique asymmetric key pair called the Endorsement Key (EK). EK is unique for each TPM — in this context, it represents the 'identity' of a TPM. A privacy CA knows the public EK either a priori or the TPM's manufacturer provides a so-called endorsement certificate which can be accessed by the privacy CA. (The endorsement certificate binds the identity to the TPM.)

[39] MATRIX is a recent framework to provide MPC system-as-a-service. A current implementation can be found at https://github.com/cryptobiu/MATRIX.

The TPM v1.1b attestation process operates as follows. During the booting process of a computer system, the executing software is measured using a cryptographic hash function (*e.g.* SHA-2). In a TPM, the TPM_EXTEND operation extends the measurements into special PCR memory registers. During attestation, the TPM is invoked with a TPM_QUOTE operation, which, in turn, returns a signed report of the PCR registers. Each report is signed by a freshly generated secondary asymmetric key-pair called an Attestation Identity Key (AIK). To ensure privacy in reporting, the TPM sends the public part of the AIK, signed by the EK, and the endorsement certificate to the privacy CA. The privacy CA asserts the validity of the inputs and issues an attestation identity certificate which is sent back to the TPM. The attestation identity certificate may be used to validate the signature of the TPM_QUOTE operation. This process allows a TPM to be able to authenticate itself with the use of the certificate. Further, this permits the detection of rogue TPMs via a blacklisting approach of a privacy CA — the CA either maintains a list of rogue TPMs (identified by the EK) and rejects requests by those; or if too many requests are queried by a specific TPM it can add this EK to the blacklist.

Direct Anonymous Attestation. The DAA protocol was introduced in 2004 by Brickell *et al.* [81] as a successor to the TPM v1.1 attestation protocol. The original protocol evolved over recent years due to security vulnerabilities and efficiency improvement: Smyth *et al.* [402] showed a security flaw in the original protocol and proposed a fix. Brickell *et al.* [83] improved on the original protocol by using symmetric pairings, rather than RSA. Chen *et al.* [108, 109] showed further improvements by moving from a symmetric to an asymmetric setting. Unfortunately, Chen *et al.* [107] showed that this improvement is insecure. To address these vulnerabilities, Chen *et al.* proposed a new adaption [110] based on elliptic curve cryptography, which is implemented in both the EPID standard [3, 4] and the TPM 2 standard [210].

Among other improvements, the DAA protocol mainly improves the privacy of a prover (*i.e.* the attesting TPM) without the need for a privacy CA. The DAA protocol was adopted by the TCG for the TPM v1.2 and TPM v2. (The v1.2 standard [206] requires DAA, while the v2 standard [210] requires ECC DAA.) DAA can be seen as a group signature scheme, however, with the feature that a signature can not be opened by any of the participants (thus, anonymity is not revocable), but the capabilities to detect rogue members. The protocol involves three entities: DAA members, issuers and verifiers. DAA members are TPM platforms or EPID enabled microprocessors. The DAA issuer verifies the platform during a join step and provides the platform with DAA credentials. The platform can then create DAA signatures using a AIK. Finally, a DAA verifier is able to verify the DAA credentials via a zero-knowledge proof without compromising the privacy of the proving platform.

Intel SGX Attestation. The Intel SGX instructions and the SGX suite [253, 29] include processes to attest an enclave to (a) other enclaves running on the same system (via intra-platform attestation), and (b) to outside parties (via inter-platform attestation). Each enclave contains two registers, a unique identity value, MRENCLAVE, and a value used for sealing, MRSIGNER. These SHA-2 hashes assert a certain plat-

form and enclave state. For intra-platform attestation[40], the EREPORT instruction issues a signed report of the two registers, user data and some other meta data signed by the Report Key issued by the EGETKEY instruction. While for intra-platform attestation a symmetric key is used, for inter-platform attestation[41] that can be verified outside of the platform an asymmetric key is required. In this context, SGX utilises a special enclave, devoted for remote attestation — called the Quoting Enclave (QE). The QE verifies reports of other enclaves via intra-platform attestation and signs a MAC over the reports with a device-specific asymmetric key. This output is referred to as QUOTE. The signature scheme described is the EPID algorithm. EPID was first introduced as an enhancement of the DAA protocol for TPMs, but then adapted by the SGX suite as attestation primitive to ensure privacy of the provers. Brickell and Li introduced EPID [84] in 2007 as an adaption of DAA with enhanced revocation capabilities. In 2009, the same authors presented a new notion of EPID [85, 82] from bilinear pairings. In 2013, EPID was standardised in ISO/IEC 20008-1:2013 [3] and ISO/IEC 20009-1:2013 [4]. Finally, in 2016, EPID was introduced for Internet of Things (IoT) security [124]. Johnson *et al.* [253] further elaborate on the use of EPID within SGX.

Finite State Attestation. The FSA protocol was introduced in 2016 by Paverd [351]. FSA is a one-to-many attestation protocol that was specifically designed to address the security and performance requirements of a TPM-based TRE. In this context, a quote issued by a single TRE can be asserted by multiple verifiers without the need to run the attestation process again. The philosophy behind the FSA protocol is that after a TRE loads its software it reaches a final state in the measurement process — a state that it will not leave voluntarily. In this context, a TRE issues an ephemeral asymmetric key pair when it reaches the final state. The private key is then stored in volatile memory of the TRE system. In case of a platform reset, the key is irreversibly lost. Further, for the public key, the TRE issues a FSA certificate. The FSA certificate is used in the TLS handshake with a verifier. The TRE can thus produce a single quote with this FSA certificate to communicate with multiple verifiers. To achieve anonymity in its reporting, FSA utilises the same concept of an AIK. Further, to prevent replay attacks, FSA utilises nonces in the TLS protocol.

6.6.4 Implementation Strategies in Pseudo Code

Before diving into the details of our benchmarking, we illustrate the setting in which we performed our evaluation. We have implemented basic operations for three evaluation cases: plain, TRE and MPC. Further, we performed evaluations in two settings — in the two party case, we increased the number of operations; in the multi-party case, we increased the number of participants.

To set a base point for the comparison, we developed a plain implementation for both the two-party case and the multi-party case. The 'plain' implementation solely implements communication operations and the basic operation under test by the eval-

[40] http://www.sgx101.com/portfolio/local_attestation/
[41] http://www.sgx101.com/portfolio/remote_attestation/

Table 16: Implementation strategies illustrated in pseudo-code for our different evaluation cases: plain, TRE and MPC.

	Plain		**TRE**		**MPC**	
A, B:	input($x_{a,b}$)	A, *TRE*:	input($x_{a,b}$)	A, B:	input($x_{a,b}$)	**2-party**
A $\rightarrow$ B:	send x_a	A $\rightarrow$ *TRE*:	initiate RA	A, B:	est. GC; gen. sec. shares	
B:	eval. $y = f(x_a, x_b)$	*TRE* $\rightarrow$ A:	respond RA	A $\rightarrow$ B:	send GC; sec. shares	
B $\rightarrow$ A:	return y	A $\rightarrow$ *TRE**:	send x_a	B:	eval. GC; sec. shares	
		TRE:	eval. $y = f(x_a, x_b)$			
		TRE $\rightarrow$ A:	return y	B $\rightarrow$ A:	return y	
p_i : $_{i=0..m}$	input(x_i)	p_i : $_{i=0..m}$	input(x_i)	p_i : $_{i=0..m}$	input(x_i)	**m-party**
p_i $_{i=0..m-1}$ $\rightarrow p_m$:	send x_i	p_i $_{i=0..m-1}$ $\rightarrow$ *TRE*:	initiate RA	p_i : $_{i=0..m}$	est. GC; gen. sec. shares	
p_m:	eval. $y = f(x_i)$	*TRE* $\rightarrow$ p_i : $_{i=0..m-1}$	respond RA	p_i $_{i=0..m}$ $\rightarrow$ p_j : $_{j=0..m/i}$	send GC; sec. shares	
$p_m \rightarrow$ p_i : $_{i=0..m-1}$	return y	p_i $_{i=0..m-1}$ $\rightarrow$ *TRE**:	send x_i	p_i : $_{i=0..m}$	eval. GC; sec. shares	
		TRE:	eval. $y = f(x_i)$			
		TRE $\rightarrow$ p_i : $_{i=0..m-1}$	return y			

* We assume that A trust the *TRE*. Same applies in the m-party setting.

uation use case. (Here, 'plain' doesn't offer any security features to protect against malicious actors.) The TRE implementation builds upon the plain version by outsourcing sensitive operations within secure *enclaves* and establishing trust through software attestation. In our benchmarking we only use SGX-based TREs (utilising EPID protected Intel SGX attestation). The MPC implementation invokes any of the previous mentioned two-party and multi-party protocols. In our framework, the global MPC implementation can be seen as a wrapper whereby the underlying protocol is any of the previous protocols.

Table 16 presents the implementation strategies in pseudo code of our evaluations. The columns of the table represent the three different cases (plain, TRE and MPC). The rows illustrate the instructions that are sequentially executed from top to bottom. We distinguish between two settings (2-party and m-party). In the two-party setting, we consider parties Alice (A) and Bob (B) (Bob may be replaced by TRE in the TRE case). In the multi-party setting, we consider m parties and denote a single party as p_i with $0 \leqslant i \leqslant m$. (The TRE represents a single party p_i.) In all cases, we assume that the protocol starts with users who input their sensitive data (denoted by $x_{\{a,b,i\}}$) using the input interface.

In the plain case, the data is then sent to the other party(ies), the evaluation function f is computed and the result is returned to the other party(ies). Further, in the plain case, for both settings (2-party and m-party), we assume that one party (B, p_m) acts as a trustworthy server.

In the TRE case, before any sensitive data is sent, all parties invoke an attestation request (RA) to which the TRE responds with an attestation result (RA). We assume the parties accept this result and continue as explained in the plain case with the difference that the functionality is computed in secure enclaves and the communication is protected (with regards to confidentiality and integrity). The TRE represents a party and acts as an intermediary to all other parties.

In the MPC case, dependent on the underlying protocol, either garbled circuits (GC) or secret shares (sec) are generated, shared and evaluated. The functionality is computed depending on the multi-party protocol.

6.6.5 Performance Evaluation

Scenario. To simplify our benchmarking of the selected protocols, we consider that the participating parties are local — running on the same computational unit. In this context, the parties are represented as a separate user-space process, whereby all processes are running on the same untrusted party. This is an appealing feature for cloud computing to reduce costs, for example, for separation and to save communication costs between the parties. (We note that in general the participants are distributed over several computational devices — the common functionality is computed by sharing information with each other.) Distributed computing implies a communication overhead, whereby every participant needs to reserve some of their resources for the computation of the function. When looking at resource-limited IoT devices, the possibility to outsource the computation to a cloud based high-end server — which, in turn,

computes the functionality and returns the result — may result in a more scalable and efficient approach.

In our scenario, each participant invokes a user-space process on the untrusted server. In this context, the user process abstracts the party — in other words, it acts as a representative of the party. Then, to collaboratively compute the functionality, the mutually distrustful parties interact in a multi-party protocol[42]. In our benchmarking suite, each party is invoked automatically and provisioned with a random input value (within a range of 4 bits), further used as random input value for the evaluated function. The party acting as a server (in the plain case and in the TRE case) are similarly represented as user-space processes. To enable communication between the server and participant processes, socket implementations are used.

Generally speaking, the following performance measurements can be applied via the previous scenario (local setting) as well as in a distributed setting. However, when remote parties are involved, the communication overhead over the network must be taken into consideration. We argue that this should be a 'somewhat' constant factor (within a range, given random network delays), but should be irrelevant when evaluating the 'interesting aspects' (computational steps of the protocols) for our comparison.

Threat Model. To better illustrate our selected scenario in which we perform our benchmarking, we consider two types of adversary: (a) a malicious cloud provider and (b) a malicious outsider. We assume that a malicious cloud provider has access to both hardware and software of the computational device (*i.e.* the cloud server), while a malicious outsider has access only to the software. We omit availability attacks (in other words, attacks preventing guaranteed output delivery of the protocol), since these attacks are nearly unavoidable — an adversary, who controls the operating system (and consequently, controls the scheduling of processes) may always be able to exclude processes necessary to run the multi-party protocol. Attacks targeting the privacy of the input values are avoided via the previously introduced two-party and multi-party protocols. In the case of the SGX-based TRE, a rogue operating system and any other possibly malicious application is detected in case it tries to disturb the protocol (via the underlying SGX primitives, which guarantee confidentiality, integrity and freshness). For a reminder of the privacy guarantees of the protocols we refer to Table 3.

Selected Benchmarking Operations. We evaluated the performance in the context of basic data analysis operations represented in Table 17. These operations can be classified according to their operational categories (Arithmetic, Boolean, comparison and composite). Further, they can be classified dependent on their applicability to two-party computation (2-party) and multi-party computation (m-party). In the case of two-party computation, we increased the number of operations, while, for multi-party computation, we increased the number of participants.

[42]Here, and, in the following text, when we write multi-party protocol, we mean this applies to the two-party protocols and the multi-party protocols, respectively

Table 17: Basic data mining operations used in our benchmarking.

	Arithmetic	Boolean	Comparison	Composite
2-party	add mult	and xor	max min equaltity	count
n-party	sum product	and xor	max min	count average

Table 18: Software versions and references of the two-party and multi-party protocol software used in our benchmarking.

Name	Version	Implementation Language	Protocol Language	Reference	
Fairplay	v1.0	java	SFDLv1	[1]	**2-party**
Tasty	v0.1.3	python	tastyl	2	
MightBeEvil	v1.0	java	java	3	
FairplayMP*	v1.1	java	SFDLv2	4	**n-party**
Sharemind	v2015.12	C++	SecreC	5	
VIFF	vtip	python	python	6	
SEPIA	v0.9.1	java	java	7	
SPDZ	v1.0	java	Arithm. circuit	8	
SCAPI	v2.4.0	C++/java	C++/java	9	

* FairplayMP was rewritten by the author to be executable in a local setting

1 http://cs.huji.ac.il/project/Fairplay/Fairplay/Fairplay_Project.tar

2 https://github.com/tastyproject/tasty

3 http://www.mightbeevil.org/

4 https://github.com/FaiplayMP/FairplayMP

5 https://sharemind-sdk.github.io/

6 http://hg.viff.dk/viff/archive/tip.zip

7 http://www.sepia.ee.ethz.ch

8 https://github.com/vitorenesduarte/spdz

9 https://github.com/cryptobiu/libscapi

Limitations and Selection Decisions. In Table 18, we list the software versions of the protocols we evaluated. The table is split according to the two-party and multi-party protocols. Apart from listing the concrete protocol version and referencing the obtained source code location, we list the implementation language and protocol language of each framework. Interestingly, two thirds of the implementations are based on java, while only two are based on python and C++. About half of the implementations defined their own protocol languages (typically based closely on the implementation language).

The benchmarked version of our SGX-based TRE is a scaled down modification of the description in Section 6.3.3. Concretely, to allow a fair comparison between the two-party / multi-party protocols and the SGX-based TRE we dropped the requirement of release privacy. This means that the SGX-based TRE utilises only the SGX

primitives to establish trust and a trusted communication channel. The TRE does not perform 'privacy-preserving' computations in the form of using privacy-preserving release algorithms (such as the GUPT based differential privacy implementation). The underlying SGX primitives are based on the SGX Linux version 1.6[43]. SGX is implemented via Intel microprocessor code instructions and offers APIs in C and C++ to implement (two-party, multi-party) protocols.

While we have introduced multiple attestation protocols for completeness and to show the historical developments, we only include the SGX attestation protocol in our benchmarking. The reasoning behind this decisions are: (a) in our benchmarking we focus on SGX-based TREs, thus only SGX attestation is available, and (b) Paverd covers a performance evaluation of TPM-based attestation protocols (we refer the reader to [351] for an evaluation of TPMv1.2 attestation and FSA attestation). Further, SGX attestation includes the EPID protocol, thus covering the DAA protocol (which also includes EPID).

Nevertheless, in the set up to our benchmarking we encountered multiple obstacles that prevent us from evaluating some of the previously discussed protocols. To summarise, the following list of obstacles apply to most of the evaluated protocols.

(a) **Common Functionality:** The lack of support to implement common functionality hinders our benchmarking and the development of applications to perform a proper apples-to-apples comparison between the protocols. As a consequence, instead of evaluating use cases such as those listed in Section 5.3 we were forced to benchmark only basic data analytics operations with the prospect to combine such operations to build complex applications.

(b) **Common Interpreter:** Table 18 lists the protocol languages of each framework. As can be seen, each of the frameworks support their own way of interpreting a user's input and the design of a protocol (which abstracts the functionality that is computed by the protocol). A lack of a common definition hinders the development of a set of common use cases to evaluate the frameworks.

(c) **Requirements of OS:** Most of the frameworks have dependencies on other libraries and specific requirements of the operating system. These dependencies harden usability and flexibility of the framework — it proves to be a tedious and laborious process to ensure that all dependencies are satisfied.

(d) **Undocumented Installation Guide:** Following from the previous point, most of the selected frameworks are merely proof of concept implementations of academic research. As a result, the frameworks only support a specific set of operating systems and porting to other versions or complete different operating systems is a cumbersome process. What is more, some of the frameworks miss a detailed installation guide and miss listing dependencies of used libraries. It remains often a trial-and-error process to install and run those libraries.

(e) **Lack of Wikis, Tutorials, APIs:** A lack of guidance in the form of tutorials and APIs hinder the integration of these frameworks into benchmarking frameworks and existing applications. Missing step-by-step tutorials require a long training

[43] https://01.org/intel-software-guard-extensions/downloads/intel-sgx-linux-1.6-release

time to get familiar with the frameworks — thus, significantly preventing the adaptation of these frameworks by non-experts.

(f) **Undocumented Code:** Most of the frameworks are implemented as proof of concept, thus the code quality is not at a production level and the frameworks are often only sparsely documented. These limitations hinder the understandability and use of the frameworks in applications for privacy-preserving data analysis.

(g) **Broken Libraries:** Although some frameworks claim to support certain operations, some miss the implementation of these functionality. This proves to be extremely frustrating for software developers that rely on these operations when using the framework in their applications (or, for us, when trying to benchmark those operations).

The previous listed limitations and some limitations specific to some protocols prevented a straightforward apples-to-apples comparison between the protocols. Hastings *et al.* [222] recently presented similar results in their systematisation of knowledge of secure multi-party frameworks. In their findings, they critisise the lack of documentation, which comes with most of these 'proof of concept' implementations, and advocate standardisation among the interfaces and to promote benchmarking between the frameworks.

More concretely, in our analysis, the Fairplay and FairplayMP framework throw an out-of-memory error when using large Boolean circuits (*e.g.* for circuits exceeding 4 bit values and over 200 operations). Tasty and VIFF exceeds our benchmarking time limit[44]. For certain operations (*e.g.* logical AND, max, min) VIFF throws a not-supported and not-implemented exception.

These limitations forced us to bound the maximum number of operations that we evaluate in our benchmarking to 1000 operations and the maximum number of participants to 1000 participants, respectively. The Boolean circuit size limitations of the Fairplay and FairplayMP framework forced us to choose input, output and intermediate values within a range of 4-bit values.

Due to the lack of common functionalities and a common interpreter, we decided to restrict our benchmarking to basic data analysis operations. However, we argue that more complex use cases can be built merely by combining these basic operations. For example, an auction scheme consists of sum and max operations, a voting scheme consists of compare, count and max operations, and set intersection schemes comprise compare and count operations (see Table 14 for a detailed list).

For our analysis we have selected two protocols from each category (two-party and multi-party) as highlighted in Table 18. Other protcols have been rejected because they support only very limited specific use cases and require specific coding requirements (for example, implementations on circuit level). The long training time to get the necessary familiarity with the framework (dependent on any given previous experience and knowledge in software development and with multi-party schemes) was a factor which led to our selection decision. Consequently, we rejected frameworks which exceeded a training time of more than 3 days (for an average software devel-

[44] To simulate real-world conditions, we limited the pre-computation and online phase to, at most, 10 minutes.

oper, concretely, the author) to make informed changes in existing code / establish new protocols.

The following points list specific reasons why we rejected the two-party and multi-party protocols:

- **MightBeEvil:** Supports natively only four use cases (as listed in the description of MightBeEvil). Building generic operations is possible with significant software development costs and hindered by the sparsely documented framework.
- **SPDZ:** Supports only Arithmetic circuits as protocol language. Building generic operations are not developer friendly and require long training times to develop or convert existing use cases to this kind of input.
- **SEPIA:** Offers multiple pre-installed sample configurations that cover a good range of application scenarios. Yet, in our context, we weren't able to adapt the framework without the need to spend significant training time to our benchmarking needs.
- **SCAPI:** The framework is built to provide building blocks to create your own 'specific' multi-party protocols. In our benchmarking, we evaluated 'generic' operations which cover a plug-and-play implementation (in other words, a simple language in which a software developer can write functionality).
- **Sharemind:** While we managed to get the Sharemind framework running in the provided virtual machine, we got errors when installing the SDK. After several tries to resolve the issues, we decided to skip the evaluation of Sharemind. (Evaluating Sharemind in the virtual machine wouldn't allow us to compare it to the other frameworks without an unknown overhead.)

As mentioned before, we acknowledge that the setting in which our benchmarking has been undertaken doesn't cover all possible scenarios in which multi-party schemes may be applied. The reader might mistakenly get the impression of a limited approach and limited performance of the selected protocols for our evaluation. The choices for selections were outlined previously — here, we want to emphasise that the concern in our benchmarking is to evaluate protocols capable of solving generic problems in the MPC domain (focusing on efficiency, scalability and ease-to-use for non-experts). Only a few frameworks satisfy these restrictions (some of which might be considered 'old'). Our results should be seen from this perspective.

Measurement Setup. All measurements were performed on a Dell OptiPlex 7040 equipped with a SGX-enabled 64-bit Intel Core i7-6700 (3.4 GHz) quad-core processor and 32 GB DDR4 RAM running Ubuntu 14.04. We used java in version 1.8.0.101 and python in version 2.7.6. The benchmarking suite was written in python and is publicly available here[45]. The suite utilises python's time.time() function as time source for the measurements. To avoid unnecessary context switches between user-space processes and potential other artifacts, each reported result consists of the average of 10 single measurements.

[45] http://users.ox.ac.uk/~kell4062/software/benchmarking_suite/

Table 19: Results for the benchmarking of the two-party protocols for the operation add. The reported results are stated in seconds.

Protocol/ # of operations	10	100	200	300	500	700	1000	10k	100k
Plain	0.021	0.020	0.021	0.021	0.021	0.021	0.021	0.029	0.28
TRE	0.056	0.056	0.057	0.057	0.058	0.058	0.058	0.069	0.30
Fairplay	0.50	1.35	-*	-	-	-	-	-	-
Tasty	0.43	3.45	9.28	18.25	44.97	84.34	169.49	-**	-

* Throws an out-of-memory error for 200+ operations.

** Exceeds timing limitation of our benchmarking framework.

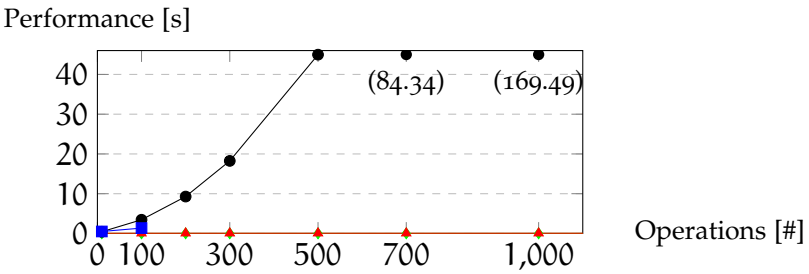


Figure 41: Comparison in the two-party setting for the operation add. Included two-party protocols: Plain(—♦—), TRE(—▲—), Fairplay(—■—) and Tasty(—●—).

Results and Discussion. For our evaluation, we have designed a benchmarking suite written in python, which automatically invokes any of the previously introduced protocols — for each party it provisions the protocol with a random input. Every party is represented as its own user-space process, that operates in a local setting on the same computational unit — following the specification of our scenario and threat model.

The benchmarking operates according to the previously introduced implementation strategies (for the cases of plain, TRE and MPC). Often, these protocols have two phases: (a) a pre-computation phase — comprising of, for example, key generation, key exchange, generation of secret shares and generation of garbled circuits; and (b) an online phase — 'the actual interactions of the secure protocol'. In our benchmarking, we evaluate only the 'online' part of the protocol. (The 'offline' pre-computation part can be done in advance for some functionalities and are thus independent of the actual secure protocol. Thus, we exclude this part from our evaluation.) To be clear, we do include remote attestation in the measurements of the SGX-based TRE, because a change in the functionality changes the attestation parameters.

From our evaluation we attained the following results, whereby Table 19 presents the measurements for the add operation in the two-party case. The results include the protocols of plain (reference point), TRE, Fairplay and Tasty. The values in the table are stated in seconds. For two-party protocols, we fixed the number of participants to two and increased the number of operations from 10 to 100k.

When we interpret the results of the two-party evaluation, we can see that the plain reference point and the TRE implementation are nearly constant up to 10k operations.

Table 20: Results for the benchmarking of the multi-party protocols for the operation sum. The reported results are stated in seconds.

Protocol/ # of participants	10	100	200	300	500	700	1000
PlainMP	0.015	0.12	0.24	0.35	0.59	0.82	1.19
TREMP	0.142	1.09	2.06	3.10	5.16	7.23	10.40
FairplayMP	2.32	6.84	12.55	18.77	33.01	50.24	73.42
VIFF	3.66	19.83	-*	-	-	-	-

* Exceeds timing limitation of our benchmarking framework.

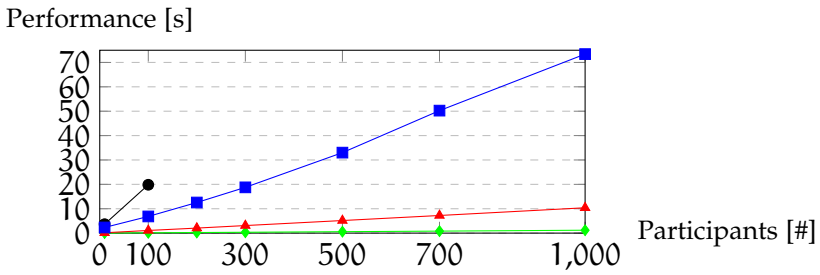


Figure 42: Comparison in the multi-party setting for the operation sum. Included multi-party protocols: PlainMP(—♦—), TREMP(—▲—), FairplayMP(—■—) and VIFF(—●—).

Any performance drawbacks of the TRE are mainly influenced by the remote attestation (which accounts for approximately 50ms [273]). Both Fairplay and Tasty sustain significant performance losses if the number of operations is increased above 200 operations. This can be seen clearly, since Fairplay is only scalable to run 4 bit operations up to a bound of around 100 operations, before the precomputed circuits throw an out-of-memory exception. Similarly, Tasty invokes our timing limitation when operations exceed 1k. This behaviour clearly shows that generic two-party protocols are not scalable for more complex applications. Further, Figure 41 illustrates the previously presented results. (The non-scalability of the two-party protocols can be clearly seen.)

In Table 20, we present the results of the operation sum for the multi-party case. The results include the protocols of plainMP (reference point), TREMP, FairplayMP and VIFF. The values in the table are stated in seconds. For multi-party protocols, we fixed the number of operations to one operation per party and increased the number of participants from 10 to 1k.

In the multi-party case, the obtained results are similar to the two-party case. The measured values for the threaded plainMP and TREMP implementations increase slightly due to the rising number of participants. Again, for both protocols, FairplayMP and VIFF, the obtained values increase significantly faster — approximately with a rate of 6–7 times in our evaluation interval. While FairplayMP scales up to 1000 participants (although, with significant performance losses), VIFF invokes our timing limitation exceeding 200+ participants. Again, we can conclude from the observed behaviour that generic multi-party schemes are not suitable for more complex applications involving an ever-increasing amount of participants. Figure 42 illustrates the

performance measurements of the protocols in the multi-party domain. (The insufficiency of the multi-party can be clearly spotted.)

In addition to the previously discussed data analytics operations, we evaluated the protocols with the operations as listed in Table 17. Interestingly, the results of these operations are very similar — within the standard deviation of our measurements. It is therefore the case that we omit presenting the results. These findings imply that the evaluated operations are independent (in the context of our benchmarking) of the core operations of the two-party and multi-party protocols. In other words, we can conclude that the 'evaluated operations' do not influence the overall execution time of the protocol in a (from our benchmarking tool) measurable manner.

6.7 A Fair Comparison: TREs vs. MPC

In this chapter, we have explored the traditional 'cryptographic approach' of multi-party computation where techniques typically rely on mathematical hardness assumptions, information theoretic results and constructions which 'dismiss' the need for a trusted third party in multi-party settings. The other approach that we have explored is based on Trusted Computing techniques, which build upon trusted components that achieve secure computation by proving their state to each other.

In this context, Hastings *et al.* [222] refer to MPC as: » *MPC can be viewed as a cryptographic method for providing the functionality of a trusted party — who would accept private inputs, compute a function and return the result to the stakeholders — without the need for mutual trust ...* « [222].

What is often missing in the design of secure communication systems is that MPC components represent only *one* of many components in such systems. As a direct consequence, trust is still needed, for example, for: (a) the implementation of a MPC protocol; (b) the environment in which the protocol is executed [178]; (c) components the protocol interacts with, depends on, or communicates with; and (d) with the inputs supplied by the potentially malicious parties. In this context, a MPC component enhances security and privacy guarantees — one could say it requires less trust or is 'trustless' (compared with other systems) — but, only seldom it completely replaces trust in other components.

In the context of Trusted Computing (TC), trust is one of the major building blocks for any component and a trustworthy system consists of a chain of trusted components. So called 'roots of trust' resemble minimal structures a user must trust — it's the foundation upon which other components construct secure complex systems. Andrew Martin [303] argues that: » *Trusted computing implies a re-design of systems architecture in such a way as to support its factorization into relatively discrete components with well-defined characteristics. This permits, in particular, rational decisions based upon reasonable expectations of behaviour ...* « [303].

In the following sections, we compare both approaches in a fair manner. Our comparison is based on different categories, which are valuated and esteemed by both academics and practitioners. However, our assessment should not yield a recommen-

Table 21: Trust comparison: TREs vs MPC

Category	TREs (SGX-based)	MPC
Mathematical Hardness Assumptions	✓	✓
Implementation (Protocol)	✓	✓
Implementation (3rd party libraries)	✓	✓
Roots of Trust	✓	~
Operating System	-	✓
Computer System (CPU)	✓	✓
Computer System (I/O, RAM, Hard drive, Network)	~	✓
Computer System (System level software)	~	✓
Orchestration (3rd party software)	~	~
Hardware Host (3rd party provider)	~	✓
Parties Input	-	-
Other Parties (Hardware & Software)	~	~

✓(Required), ~ (Partially Required), - (Not Required)

dation of either of the two approaches, but present insights for the reader to make an informed decision (based on the readers own requirements).

Direct comparisons between those two approaches are limited in the literature. We refer to Choi and Butler [114] for a more detailed look between those approaches.

6.7.1 Trust

An Environment of Trust. Most secure systems rely on multiple components which are designed and build by multiple stakeholders. Evans *et al.* [178], recognize this issue of multi-party systems: » *... protocols are executed by complex software and hardware, incorporating thousands of components provided by different vendors and programmers* « [178].

This example shows that it is essential to trust other components. It is unlikely that a whole software system is built by only one trusted entity. As such, end-users usually do not have much choice, but to make decisions to trust and engage with certain systems [178].

Trust Components of TREs and MPC. In more detail, in Table 21, we lay out the different components that a user is typically required to trust when engaging with TREs and multi-party schemes.

In the table, a category is ticked as 'required' if the technology is mandatory for secure execution of a protocol; a category is 'partially required' if the technology must be partially trusted; and a category is 'not required' if the technology does not influence the secure execution of the protocol.

As a result, we can derive that both approaches have similar components which they must rely on. As emerging patterns, we can see that TREs ground their trust

mainly on hardware, whilst MPC schemes ground their trust mainly on information theoretic hardness assumptions.

Hardware-based Trust. Hardware-based trust relies typically on three types of root of trusts [303]:

(a) root of trust for measurement,

(b) root of trust for storage, and

(c) root of trust for reporting.

A root of trust is the lowest level of a component from which trust originates. For SGX-based TREs[46] the *root of trust for measurement* can be defined as a combination of the implementation of the micro-code command ECREATE (which initialises the enclaves identity MRENCLAVE) and the 256-bit SHA-2 initialization algorithm [310, 128]. The *root of trust for storage* depends on the binding or sealing operation and is a combination of the enclaves identity MRENCLAVE (sealing), the enclaves sealing identity MRSIGNER, the implementation of the micro-code operation EGETKEY (to retrieve sealing/binding key) and the dedicated encryption algorithm (chosen by the software developer)[47] [29, 128]. Further, for reporting it can be differentiated from the intra-platform attestation and the inter-platform attestation. For intra-platform attestation[48], the *root of trust for reporting* is a combination of the micro-code command EREPORT, EGETKEY and Intel's implementation of its AES128-CMAC [164] message authentication code algorithm to issue a REPORT structure [29, 128], which can be used to verify the enclave's state. For inter-platform attestation, it is further required to trust a special reporting enclave, called *Quoting Enclave* [253, 126, 29] and Intel's implementation of the EPID protocol [124, 253].

Trust Based on Information Theoretic Results. Trust in information theoretic hardness assumptions requires, for example, a software developer to trust the implementation and hardness assumptions of these primitives. The basic building blocks which most MPC protocols are build upon are: (a) garbled circuits; (b) oblivious transfer; (c) secret sharing; and (d) homomorphic encryption. For *garbled circuits* based protocols we must trust a combination of *oblivious transfer* [365] and a symmetric encryption scheme such as AES [333]. For *oblivious transfer* [365] based protocols we typically must trust a public-key encryption scheme such as the RSA algorithm [374]. For secret sharing based protocols [395, 63] we must trust the implementations of Shamir's secret sharing [395] algorithm based on polynomial interpolation of Lagrange polynomials and finite field theory; or Blakley's algorithm [63] based on the intersection of points in the hyperplane. For homomorphic encryption based protocols, we must trust the homomorphic encryption scheme (*e.g.* RSA [374], Pallier cryptosystem [345], ElGamal cryptosystem [172] and Gentry's full homomorphic cryptosystem [191]).

[46] See Part II of Chapter 2 for a more detailed description of the roots of trust and SGX operations.
[47] http://www.sgx101.com/portfolio/sealing_primitives
[48] http://www.sgx101.com/portfolio/attestation_primitives

User-Objective Trust. From a system developer perspective, the developer must trust that for:

- **SGX-based TREs:** Intel is developing and shipping genuine Intel CPUs and support the Intel SDK without any backdoors and security vulnerabilities that can be exploited at a later stage by an adversary.
- **MPC:** the library developer didn't implement the protocol incorrectly; and the protocol designer didn't overlook any design vulnerability in the MPC protocol and security proof.

From a user perspective, for all of the previous protocols a user must trust: (a) the protocol specification, (b) the protocol implementation, (c) any third party libraries (used to implement the protocol) and (d) any information theoretic results on which those protocols are based upon / rely on.

6.7.2 Security

Given the previously discussed trust assumptions, we briefly compare the security of both approaches. We note that it is often not possible to perform an 'apples-to-apples' comparison, since these types of protocols differ significantly and finding a common 'measure' to assess their security may be a challenging task. In addition, our results are based on the comparison of two 'concepts' and not concrete instantiations of a protocol.

Threat Models and Overview. Given this context, any security guarantees depend on a threat model; we compare between: (a) malicious; (b) semi-malicious; and (c) covert adversarial behaviour. A detailed threat model of SGX-based TREs is presented in Section 6.4 and Section 6.5. The threat model of MPC protocols are given by the underlying definition of multi-party computation (see Definition 15 or the security definitions of multi-party protocols by Canetti [95]). In addition, most multi-party protocols require the following properties: (a) *privacy*, (b) *correctness*, (c) *independence of inputs*, (d) *fairness*, and (e) *guaranteed output delivery* [290].

In Table 3 we outline further security categories of some of the previously defined protocols and their cryptographic primitives upon which they rely.

Furthermore, in Table 22 we provide an overview of the different security assumptions between the two approaches discussed in this chapter.

Underlying Primitives. The underlying primitives of SGX-based TREs are the SGX primitives and their cryptographic procedures. As a consequence, the security guarantees are based on these primitives which we have reviewed in Section 6.5.2.

While SGX relies on hardware based trust, MPC protocols typically rely on information theoretic / mathematical results and assumptions. At a core level, both approaches rely on the following assumptions, operations and functional components:

- **Randomness:** SGX and many MPC schemes rely on the generation of strong randomness as a basic building block for the protocol [239, 305].

Table 22: Security comparison: TREs vs MPC

Category	TREs (SGX-based)	MPC
Trust Model	Trust in Hardware	Math. Assumptions Information Theory
Roots of Trust (RoT)	RoT for measurement RoT for storage RoT for reporting	Software impl. of protocol
Dependencies	Intel CPU Intel SDK	3rd party libraries
Categories	$\mathcal{A}_{release}$ $\mathcal{A}_{in/out}$ $\mathcal{A}_{system}$	Malicious Semi-malicious Covert
Security Primitives	SGX primitves	Garbled circuits, Obliv. transfer Secret sharing, Homom. encryp.
Potential Vulnerabilities	Side-channel leakage	Operating System Software impl.

- **Public-Key Cryptography (PKC):** SGX and many MPC schemes rely on public-key cryptographic schemes for either encryption (*e.g.* encrypting the inputs of a garbled circuit) or to digitally sign a value (*e.g.* for attestation). PKC typically builds upon the following assumptions: (a) the integer factorisation problem [59] and (b) the discrete logarithm problem [59].
- **Symmetric Cryptography:** SGX and many MPC schemes rely on symmetric cryptography for basic building blocks of hash functions (*e.g.* utilised in attestation protocols).

For both SGX and MPC protocols, the achieved security guarantees depend on the specific protocols used to implement this functionality. We refer to the SGX documentation [126, 128] and the specific MPC protocols (for example, those discussed in Section 6.6.1 and Section 6.6.2).

Adversarial Majority. Following the security definitions of Lindell and Pinkas [290], MPC protocols in which an adversary is able to compromise a threshold of $t < n/3$ (with n the number of all participants in the protocol), the properties of fairness and guaranteed output delivery can be achieved without any additional assumptions. For $t < n/2$, an additional broadcast channel is required to achieve the same guarantees. In the case of $t \leqslant n/2$, fairness and guaranteed output delivery can not be achieved.

The literature for threshold adversarial majorities is limited in the case of SGX based protocols. No previous studies (to the best knowledge of the author) have been performed with regards to threshold majority. Assuming that the attestation primitives of a party are not compromised, the party may only affect the properties of fairness and guaranteed output delivery.

Consequently, there hasn't been a mapping of the security properties of, for example, [290] to the SGX primitives (or the domain of SGX-based TREs). Although we conjecture that an adversary which may compromise a party[49] and, assuming that the attestation primitives are uncompromised, she may only participate as a normal party in the protocol (without affecting the properties of fairness or guaranteed output delivery).

Leakage via Side-Channels. All computations on a computer have a certain footprint which may leak information via a side channel to a system adversary. For MPC protocols this may happen during the setup, input, and evaluation phases. Depending on the operations, in each phase an adversary may learn certain information (*e.g.* by observing the cache of the system) which she can use to her advantage to attack a party or the integrity or privacy of the protocol evaluation. Evans *et al.* [178] further discuss different leakage trade-offs for MPC protocols that a system designer must take into consideration to cope with efficiency / privacy trade-offs.

SGX, generally, limits the threat surface by encrypting and integrity-protecting every data and code page outside of the immediate CPU level. In addition, Intel advocates to perform any confidential operations only within secure enclaves. With regards to side-channel attacks, Intel, however, leaves this as a matter of the software developer: » *Preventing side channel attacks is a matter for the enclave developer.*[50]«

Recent attacks, for example, CrossTalk [366], CacheOut [387], SGAxe [386] and those presented in Section 6.5.2 have shown that the security guarantees of SGX may be broken in this context.

6.7.3 Privacy

Release Privacy. By definition, multi-party protocols only consider privacy of the inputs of the protocol participants (see Definition 15). However, these protocols often omit to protect any leakage from the protocols results, as outlined by Definition 4. In Section 6.2.5, we provide further details and examples of this issue.

The protections offered by the SGX package also ignores release privacy. As a consequence, our SGX-based TRE prototype as introduced in Section 6.3.3 explicitly requires such protections via its personalised differential privacy layer based on the GUPT library [322].

6.7.4 Performance

We refer the reader to Section 6.6 for a detailed performance evaluation between our SGX-based TRE prototype and existing general purpose MPC frameworks. As a summary, for special use cases MPC performs reasonably well, but, for generic computation, the MPC frameworks are still lacking in performance. In this context, SGX-based TREs benefit from the execution of microcode directly within the main CPU package.

[49] A party, which participates in the SGX based protocol.
[50] https://software.intel.com/en-us/articles/intel-sgx-and-side-channels

6.7.5 Real World Considerations

Maturity. In a real-world deployment, the maturity of a technology often plays an important role. More mature prototypes and technologies promise typically well analysed security analysis and a wider range of supporting documentation and implementation examples for software developers. For cryptographic prototypes, this means that the community has been able to review these primitives and has been able to form an opinion and critically assess them.

In the context of secure computation, Archer *et al.* [38] outline a maturity hierarchy of secure computation protocols (including some of those that were evaluated in this dissertation). Their classification includes three categories: (a) academic prototype, (b) real-world deployment and (c) market ready. The protocols discussed in this dissertation, Fairplay [301], Tasty [226], and VIFF [135], are classified as *academic prototype*; Sharemind[51] [67] is classified as *market ready*. Furthermore, Unbound[52] (former Dyadic) and Partisia[53] are considered to by market-ready software packages.

In comparison, SGX was designed as a market-ready software package and includes sample source code for many operations (*e.g.* local and remote attestation), APIs, libraries, documentation and tools in form of software development kits.

Integration. Other important aspects in real-world deployments are integration capabilities of software packages. With the rise of cloud computing, a wide range of software runs in the cloud on providers such as Amazon AWS[54], Google Firebase[55] and Microsoft Azure[56]. To secure distributed computation, some providers support SGX and/or other enclave technologies. Examples include: Azure[57], IBM[58] and Google's Asylo [59] and Project Oak[60]. Furthermore, the Confidential Computing Consortium[61] fosters integration of enclave technologies and supports multiple operating systems and programming languages.

The author is unaware of any (commercially) readily-available MPC deployments and integrations in the cloud.

Support and Usability. Non-experts such as software developers and system designers require support and documentation to deploy any libraries. Furthermore, continuous maintenance of any software package and libraries is a must-have feature — to resolve issues, but also to add new features to the software package. For easy adoption, integration (as discussed before) and usability (*e.g.* sample code, documentation) must be readily-available and diverse.

[51] https://sharemind.cyber.ee
[52] https://unboundtech.com
[53] https://partisia.com
[54] https://aws.amazon.com/
[55] https://firebase.google.com/
[56] https://azure.microsoft.com
[57] https://azure.microsoft.com/en-us/solutions/confidential-compute/
[58] https://www.ibm.com/cloud/data-shield
[59] https://asylo.dev
[60] https://github.com/project-oak/oak
[61] https://confidentialcomputing.io/

SGX-based TREs in this context benefit from the continuous support and maintenance by the Intel corporation. In their offering, Intel's SGX suite [29, 310, 253, 232, 388, 128, 444] provide for software developers a software development kit [122, 127] including multiple application programming interfaces (APIs), libraries, documentations [125], sample source code and other tools.

Many MPC frameworks and implementations are proof-of-concept implementations or academic prototypes only supported by an academic research group as outlined by Archer *et al.* in their classification [38]. As such it often takes a significant effort to implement such protocols, as stated by Hastings *et al.* in their survey and comparison of recent general purpose compilers [222]: » *implementing our simple example programs in each system required significant engineering effort: we estimate over 750 person-hours* « [222]. Another limiting factor in this case is documentation and sample codes for non-experts. Hastings *et al.* advocate for improved documentation and standardisation: » *universally, the biggest obstacle when using MPC frameworks was a lack of documentation* « [222].

Standardisation. The development of industry-wide standards is an important step for any technology to reach wide-scale distribution and interoperability between systems. Standards endorse certain structures and guidelines and enable interoperability and simple substitution and replacement of protocols and software packages.

For hardware-based secure computation, the Trusted Computing Group (TCG)[62] endorses and promotes standardisation since its formation as Trusted Computing Alliance in 1999. Most importantly are the standards around the TPM [206, 207, 208, 210, 5]. Intel SGX are proprietary instruction codes built into modern Intel CPUs and closed-source. To counter that, there is an initiative of researchers of the Georgia Institute of Technology who released an emulation of SGX via the QEMU[63] system emulator called *OpenSGX*[64] [247, 266].

For traditional multi-party protocols, to the best knowledge of the author, no standard yet exists. Adoption of multi-party technology is mainly driven by the MPC Alliance[65], a consortium of multiple industrial partners and academia. Initiatives to put forward a standard for multi-party computation is on its way through two working groups: (a) the P2842 — Recommended Practice for Secure Multi-party Computation[66] by IEEE and (b) the ISO/IEC NP 4922-1 Information security — Secure multiparty computation — Part 1: General[67] by BSI.

Licensing and Distribution. The licensing model of a software package plays an important role for software and system designers in the design phase of their software. The license permits a software developer to use the software under certain conditions — outlined in the license. Further, the distribution model enables different interfaces to interact with the licensed software package.

[62] https://trustedcomputinggroup.org/
[63] https://www.qemu.org/
[64] https://github.com/sslab-gatech/opensgx
[65] https://www.mpcalliance.org
[66] https://standards.ieee.org/project/2842.html
[67] https://standardsdevelopment.bsigroup.com/projects/9020-03692#/section

In the context of Intel SGX, for development enclaves one does not need a license. But, for production enclaves — to use Intel's attestation services — one needs to obtain a commercial license[68]. Costan and Devadas in [128] remark: » *the Launch Enclave is intended to be an enclave licensing mechanism that allows Intel to force itself as an intermediary in the distribution of all enclave software* « [128]. As a consequence of this licensing model, Intel has the final control over any developed software packages that use Intel SGX technology. Yet, SGX technology is free to use for any software developer.

Traditional academic multi-party prototypes typically license their protocols under open source license, which allows the free usage of these protocols and software. Yet such software packages come with limited support and documentation. Commercially available multi-party protocols such as those offered by Cybernetica, Unbound and Partisia are closed source and are available under commercial licenses.

6.7.6 Discussion

To summarise, we want to emphasise that this 'confrontation' between TREs and MPC should not be seen as an apples-to-apples comparison. Both approaches to secure multi-party computation are complementary and, at best, enhance each other; and should not be seen as a replacement of each other.

While the techniques of Trusted Computing arose with the Internet boom in the 2000s, traditional techniques of multi-party computation were first discussed in the late 1970s. As a consequence, the traditional methods got scrutinised early and attracted a community to propose and analyse new schemes. As seen, the traditional methods focus on strong mathematically secure models and trust is based on information theoretic results. From the definition of secure multi-party computation, Definition 15, it is assumed that no single party trusts any other participant in the protocol and no one would trust a third party with their private inputs.

TREs build upon hardware-based trust models. Concretely, our SGX-based TRE prototype relies on the correct implementation of Intel's SGX microcode. In this context, trust and security is based on different primitives such as roots of trust, attestation primitives and isolated execution. Such assumptions may be arguably weaker than traditional mathematical hardness assumptions, but this relaxation enables better performance and flexibility in terms of generic function execution.

Instead of a 'comparison' the main contribution of this section is that it provides experts and non-experts with an overview of the benefits and potential weaknesses of each approach in a fair manner. To conclude, we quote Archer *et al.*: » *... secure computation is a good approach for tasks with multiple parties who trust each other to behave honestly but still need to deploy means to ensure the privacy of the computations* « [38]. Archer *et al.* emphasises that current practical approaches to secure computation require some form of trust (*i.e.* passive security) and they indicate the need for privacy of the result of the computation (in context of this dissertation — release privacy). As a consequence, both approaches, TREs and traditional MPC have their specific use cases — at best, they operate together. To end on an optimistic note, we quote Evans *et al.*: »

[68] https://software.intel.com/content/www/us/en/develop/topics/software-guard-extensions/request-license.html

Many challenges remain, but MPC (and secure computation broadly) is a young and vibrant field, with much opportunity to create, develop and apply « [178].

6.8 Summary

In previous chapters, privacy-preserving systems relied on assumptions of a system model, privacy model and adversary model — assumptions that ease the design, development and analysis of such systems. In the real-world these 'assumptions' are bothersome, not followed and invalid. It is the essence of this chapter to drop these assumptions and evaluate protocols and systems that tackle the problems of privacy-preserving data analysis and data release *without* 'restrictions'.

The cryptographic approach of secure multi-party computation drops the assumption of the existence of a trusted third party — protocols following this definition guarantee privacy to the private inputs of a set of parties that aim to jointly compute a function over these inputs. While the 1980s and 1990s saw theoretical progress and some proof-of-concept implementations, in the previous two decades more and more practical implementations for generic multi-party computation have evolved. The progress for large-scale, efficient and flexible schemes in this domain is however still insufficient: a large communication overhead, a computation overhead, the use of expensive operations, diminishing features caused by corrupted parties and foremost (also, in the context of this dissertation) ignorance of release privacy are factors which impair this approach.

An alternative approach, which aims to tackle the aforementioned challenges, was first presented by Paverd [351]: Trustworthy Remote Entities (TREs). These entities, while (in their original version) operating similar as a trusted third party can prove their trustworthiness through a process of software attestation. Paverd's TRE, formerly introduced in the smart grid, acts as an intermediary to the communication partners. Our contributions throughout this chapter (and other parts of this dissertation), extended the 'current' model of a TRE. To summarise, we introduce different architectural design models (centralised, homogeneously and heterogeneously split), extend the application scenario into the data analysis domain (focusing on distributed multi-party use cases) and introduce a novel design approach based on the Software Guard Extensions (SGX) primitives.

Paverd's prototype, based on Trusted Platform Modules (TPMs), introduced a novel attestation protocol — Finite State Attestation (FSA) — which issues one-to-many attestations, meaning one prover can attest to multiple verifiers without re-executing the whole attestation process. The SGX primitives as well as the TPM are based on Trusted Computing (TC) principles. Since the SGX primitives execute directly on the CPU, they supersede the cryptographic co-processors (TPMs) in multiple aspects including performance, utility, security, flexibility, scalability and their independence of specialised hardware.

Given these benefits, we propose a prototype of the TRE based on SGX primitives. To ensure release privacy, separate privacy mechanisms must be deployed. Since SGX-based TREs are not bound to any specific privacy mechanism our prototype supports

various variants of personalised differential privacy (PDP): provenance, value, analyst and query-based PDP. Concretely, our prototype utilises the Graphene libOS and extends the GUPT library by our novel personalised privacy driver. To further assess and reason about potential privacy properties of the TRE's release, the privacy games introduced in Chapter 3 may be utilised.

In real-world environments adversaries are not limited by assumptions placed upon them. We've presented a 'traditional' threat model and security analysis reviewing known attacks [298, 282, 188, 439, 430, 262, 45]. Importantly, extending this context, we have shown devastating real-world attacks [445, 291, 271, 87, 437, 390, 449, 411, 448, 213, 105], attacks against the SGX primitives [397, 392, 79, 321, 200, 435, 406, 306, 391, 429, 281, 225] and various weaknesses of privacy mechanisms (see above and [215]). To design secure systems, we followed up with mitigation techniques to tackle previous challenges.

In any real-world environment it is significant to recognise that performance — be it in terms of efficiency, utility, flexibility, scalability or the level of privacy — are the factors that typically count in the selection process of a software system. In this context, we test the relative efficiency of the two approaches via prototyping and perform a performance evaluation with regards to the number of parties a protocol can support and with regards to the number of operations a protocol can handle. Summarising our findings, we conclude that multi-party schemes still suffer from various weaknesses preventing their utilisation in large-scale distributed contexts. A SGX-based TRE system may be seen as an appealing alternative: the performance in the two-party mode is nearly optimal (compared to a plain implementation) and in the multi-party mode by a factor of 7 times better (compared to the closest generic multi-party scheme, FairplayMP [56]).

Finally, we critically review both approaches side-by-side and present advantages and disadvantages of the same. In this context, we learned the different trust models of the traditional approach, which is based on information theoretic results and our approach, which is based on hardware-based trust, roots of trust and isolated execution. Both approaches deploy different security and trust models — each of them having certain benefits and shortcomings. Overall, the best outcome is achieved by a combination of the two approaches. The field of secure computation is still young; many challenges remain — which leaves much opportunity to create, develop and apply.

This chapter ends Part II. In the next chapter we conclude and summarise the contributions of this dissertation. Our research has produced new insights in the field of privacy-preserving data analysis and data release. In addition to discussing these findings, we outline future directions of research which can extend and mature our approaches.

Part III

Conclusions

» *You know nothing, John Snow.* «

— Ygritte, Character of Game of Thrones

7 CONCLUSION

Contents

In this chapter, we draw conclusions from this dissertation. We summarise our contributions and findings established throughout various research projects over the last four years and presented in the previous chapters. Following this, we draw conclusions from our findings. Next, we elaborate on assumptions and limitations of our approaches. Then, we discuss future directions of research within this area. Finally, we draw personal conclusions of the journey over the last four years.

7.1 Contributions and Findings

In this dissertation, we have presented novel research contributions in the areas of privacy, trusted computing and secure multi-party computation. The dissertation is divided naturally into four parts (where the middle two parts comprise our main contributions): the first part introduces the reader to the relevant foundations, Part I introduces a novel way of understanding privacy notions via privacy games, Part II focuses on the application of privacy-preserving TREs in practice, and the last part concludes the findings of this dissertation.

7.1.1 Foundations

In Part I of Chapter 2 we elaborate on current definitions of privacy. From our findings, we concluded that establishing definitions of privacy is a complex, context-dependent, variable, inconcise and very hard task. One may conclude that finding an accurate and commonly accepted notion of privacy is impossible.

In Part II of Chapter 2 we introduce our field of application and survey current techniques to preserve privacy in this context. Our findings show that the literature around syntactic privacy is substantial, however, the literature for more recent tech-

niques such as differential privacy [159] is mostly limited to academic prototypes (and real-world deployments of such techniques are limited).

7.1.2 Part I

In Chapter 3 we present novel privacy games — games that outline the necessary steps of an adversary in order to break an associated privacy notion. Our findings include novel game-based definitions in semi-distributed settings. The games aim to provide a better understanding of privacy notions to non-experts and should foster analysis and comparisons between the privacy notions.

While Chapter 3 is purposefully theoretical to introduce the system model and definitions underlying the privacy games, in Chapter 4 we take a more practical turn and analyse the privacy games via a case study based on recommender systems. After outlining limiting factors of the approach, we show by means of the case study how the privacy games can be used to argue about privacy properties of systems. Our contributions make the first step to analyse privacy properties in such a way. Future approaches in this endeavour may leverage and optimise our findings. Our approach to analyse privacy properties should be seen as an alternative to existing approaches.

7.1.3 Part II

In Chapter 5 we present our findings of an analysis of several real-world use cases in distributed settings. Our results of the analysis are used to derive requirements for the design of secure and privacy-preserving real-world prototypes with a focus on large-scale settings.

Finally, in Chapter 6 we aim to combine the lessons of the previous chapters and present the application of an alternative approach to the current state-of-the-art techniques that are used to assure privacy-preserving computation between multiple parties. In this context, our findings from surveying current generic secure multi-party schemes present a critical analysis that shows limitations and pitfalls with respect to large-scale deployments. As a consequence, we introduce an alternative approach based on the combination of the novel Software Guard eXtensions (SGX) and personalised differential privacy extensions to the GUPT [322] differential privacy library. Our findings, dubbed SGX-based TREs, improve upon previous flexibility and utility limitations, resolve the inherent release privacy problem and are up to 7 times more efficient (based on our performance benchmarks). Furthermore, our findings include a threat model that illustrates current state-of-the-art attacks involving every aspect of a system — from high-level privacy-preserving mechanism attacks down to low-level side-channel attacks — and mitigations to tackle these vulnerabilities. Finally, performance benchmarks with respect to simplistic data analysis operations show the superiority of TREs over generic secure multi-party schemes.

7.2 Conclusions

Summarised in the previous section, this dissertation has produced multiple novel findings and introduced alternative approaches to look at privacy notions. In this section, we draw conclusions from these findings.

As a reminder, this dissertation, while in (some) parts highly technical, aims to foster understanding of privacy for non-experts and aims to present simplifications to understand the complexity of privacy-preserving systems.

In the foundational part we provided an introduction to the field of privacy-preserving data analysis systems. From the developments that we have observed within the period of this endeavour, we can conclude that there has been a significant shift from data release systems towards data analysis systems. The former typically releases 'anonymised' subsets of the database, while the latter allows an analyst to query the database and returns 'privacy-preserving' responses. Syntactic privacy systems inherently fall short due to background knowledge attacks. With the introduction of differential privacy [159] by Dwork in 2006, there has been a significant shift in the privacy community to study probabilistic mechanisms that offer proof of privacy guarantees.

Yet, at the time of writing, the deployment of differential private algorithms/techniques is mostly at the level of academic prototypes (with some limited deployment in specific real-world products[1]).

Due to the shortcomings of a vast amount of privacy notions, we aimed in Part I to provide alternative ways to understand a limited, yet expressive set of privacy notions via the representation as privacy games. While differential privacy-like notions are clearly on the rise, our privacy games focus on syntactic notions. On the one hand, while the mathematics of differential privacy are straightforward, an intuitive understanding of the notion can be elusive, as can characterisations of its relationships with, for example, anonymity, participation hiding or unlinkability. For non-experts, differential privacy may still remain a fairly complex mathematical concept that is hard to grasp. On the other hand, the vast variety of syntactic notions lead non-experts to confuse which privacy notion attains which system properties and invites misconceptions, misinterpretations and errors. Our privacy games aim to resolve these gaps to foster understanding for non-experts. From our endeavour to define 'simple-to-understand' privacy games, while staying generic to cover an expressive set of privacy notions and establish a system model that is able to abstract a wide array of real-world systems, we are proud to have presented first steps. While some limitations still remain (discussed in the next section), we conclude that our approach should be seen as an alternative to existing ways to express privacy notions.

From an analysis of various real-world use cases that focus specifically on privacy concerns of multiple parties in distributed environments, in Part II, we were able to derive a subset of common requirements and constraints. These constraints are incorporated in these kinds of systems and are paramount to resolve when designing secure, efficient and privacy-preserving systems. While many legacy privacy-preserving systems rely on a centralised trusted third party, considering distributed systems with

[1] For example, Apple [241] uses differential private methods to get statistics of emoji usage.

mutually distrustful participants opens up a wide array of new requirements and constraints. Our analysis should make application designers aware of these constraints and our contributions in Chapter 6 point them in the right direction with respect to how to tackle those challenges.

Concretely, Chapter 6 addresses approaches to input privacy concerns in distributed systems. While a wide array of literature makes assumptions on the adversary, we address these concerns. Secure multi-party schemes, while providing strong security guarantees, dismiss any claims of release privacy. Further, our analysis of a range of generic schemes indicates that the maturity of secure multi-party schemes remains limited. In this context specifically, concerns about real-world scalability, efficiency, utility and flexibility of such systems are questionable (and, at the time of writing, at most attainable by a limited array of specialised use cases). As a direct consequence, we adopt an alternative approach introduced by Paverd [351]. Our tweaked version (SGX-based TREs) retains better efficiency, utility and flexibility as well as strong privacy guarantees[2]. From our security analysis we can conclude that, while recent attacks introduced devastating breaks of both Intel CPUs[3] and the SGX[4,5], a careful design of privacy-preserving systems is paramount. Further, our performance benchmarks leave us to conclude that TRE systems indicate a clear superiority in terms of efficiency due to the direct execution of instruction on the CPU and limited additional computations.

7.3 Assumptions and Limitations

Having stated the contributions of this dissertation and drawn conclusions of the same, we now state assumptions and limitations of our approaches.

In Part I of Chapter 2 we surveyed the literature surrounding privacy definitions and mainly follow the legal approach to privacy by observing privacy definitions such as those presented in the GDPR [177] and HIPAA [1]. This is mainly because, de jure, privacy-preserving practices have to follow laws and comply with a country's legal framework. De facto, many of these privacy definitions are in the hands of the designer and implementors. Our assumptions and limitations here-within are following the legal path. Given the broader context of privacy, an approach from a different viewpoint may lead to other, potentially more well formed, definitions of privacy. A more inclusive, but global, view on definitions of privacy may ameliorate some limitations. (A broader view is out of scope of this dissertation, but may be studied in future work.)

In the chapters pertaining to Part I, we introduced privacy games, that, by their definition, follow certain rules and constraints. Some of these constraints are inherently given by the system, privacy and adversarial model. Some other constraints and limitations of the privacy games are discussed in Chapter 4. To reiterate, to apply

[2] Due to the inherent requirement to trust Intel hardware, these guarantees may be arguably weaker than secure multi-party schemes.
[3] Spectre [271], Meltdown [291] and Zombieland [390]
[4] Foreshadow [87, 437]
[5] Applying microcode and software patches and other mitigation techniques prevents those attacks.

the privacy games it requires the release of multiple elements, whereby each outcome element must have a relationship to a user (via a mapping to the user identifier). In this context, the privacy games apply to syntactic privacy notions and target privacy systems that release 'anonymised' releases (as opposed to the model of probabilistic privacy notions with query access). What is more, our analysis is based on logical reasoning and informed argumentations, henceforth, limiting our approach. On the contrary, dedicated formal proofs often make assumptions, and if blindly trusted, overseen errors can be devastating (*e.g.* [242]). In this context, we may argue that well formulated arguments may be seen as a strong alternative to formal proofs.

In Chapter 4 our privacy games are applied to a case study on recommender systems. While recommender systems are timely, simple to understand for non-experts and fit seamlessly with our system and privacy model, other use cases may show different applications of our privacy games. In this context, our choice of use cases may be seen as a limitation of the approach — we argue that a thorough investigation and application of our privacy games via a multitude of use cases can reveal interesting artifacts that may be only prevalent in specific scenarios.

In the chapters pertaining to Part II, we study privacy constraints in 'real-world' distributed environments. In this context, from our analysis of a range of distributed use cases we were able to derive a set of requirements and constraints. While our analysis was bound by time and resource constraints to a limited set, a wider array of use cases may have disclosed other constraints or lead us to derive different conclusions. While this will be true for any analysis, our use cases were selected to reflect an expressive subset to extract a wide range of real world features.

Finally, in Chapter 6 we started with an analysis of generic secure multi-party schemes. Our selection of generic protocols may not cover the newest developments in the research of such schemes and leave the reader with the mistaken impression of a limited approach. Yet, we study the applicability of privacy to *any* kind of computation, thus ruling out a lot of schemes designed to support a limited array of use cases. Further on, as can be seen in our advanced threat modelling, the selection of the Software Guard eXtensions may introduce side-channel vulnerabilities of our alternative approach, that was proposed to enhance utility, flexibility and privacy in distributed systems. Similarly, the choice of SGX limits us to Intel CPUs for the TRE (communicating parties with the TRE do not have these limitations). While these limitations are more prevalent in the personal computer market, it is merely a limitation for server infrastructure[6]. Furthermore, our benchmarking is solely based on a execution time analysis, thus limiting our analysis. A more in-depth analysis involving other performance factors such as utility, flexibility and scalability may lead to different results. (Although, we can argue that from a time analysis we can predict several of these factors, for example, the scalability of such schemes.)

[6] https://www.statista.com/statistics/735904/worldwide-x86-intel-amd-market-share/

7.4 Future Directions

While our contributions presented in the previous chapters have the potential to enhance the way of understanding privacy for non-experts, we hope that this dissertation may be seen as a starting point for fruitful discussions to *simplify*, *analyse*, *support* and *promote* privacy. Simplifying the understanding of privacy notions is a continuous, open-ended task — in this section, we discuss open problems and possible avenues for future work.

In Part I of Chapter 2, we studied definitions of privacy. While we believe that it is impossible to find *a* commonly accepted notion of privacy, we believe there is a huge potential to establish a framework that enables multiple stakeholders to reason and argue about privacy properties and privacy notions. Such a framework must be self-explanatory, yet, expressive to cover a wide array of systems. This can only be a joint effort including the multitude of legacy notions, while remaining open to novel contributions. Given such a framework, privacy advocates, software developers and legal scholars can discuss privacy on the same 'level' with the same expressiveness.

In Chapter 3, we studied the definition of privacy games. This approach is adapted from the cryptographic community where it is used to analyse cryptographic proofs in a simple way. While our adaption to privacy notions is novel, it is also limited and requires further analysis. Further advancements in this area may include other system, privacy and adversarial models. Other avenues to improve this alternative approach include the extension of the games to fully distributed systems (*e.g.* privacy games between systems), enhanced policies for the selection of a 'fitting' privacy game, and the study of privacy primitives (counterparts of cryptographic primitives) that ensure the guarantees stated by the privacy notions.

In Chapter 4, we analysed our privacy games and studied their application to real-world systems. In this context, possible avenues for future work include a formal analysis of the games to 'better' argue about relationships between the games (*e.g.* implications, equivalence, order). Further, since we applied these games only to one case study, utilising these games to study privacy properties of a wide range of use cases may reveal new interesting artifacts.

In Chapter 5, we derived requirements and constraints for large-scale prototypes. While our analysis included a fair set of use cases, considering a wider range of use cases may reveal more detailed insights to derive new constraints or lift existing constraints. Possible other avenues in this direction are to specify constraints for specific types or classes of privacy-preserving systems to allow a more fine grained specification of privacy-preserving systems.

In Chapter 6, we investigated an alternative approach to secure multi-party computation schemes. While at the start of this project SGX was a novel technology, it has received a lot of research attention over the last four years. In the context of SGX-based TREs, studying the implications of fully distributed SGX systems may lead to interesting results, as proposed in our architecture model. Other avenues include the adoption and comparison of various release privacy techniques (on top of a SGX-based TRE), extension of personalised differential privacy modes, and the utilisation of novel

SGXv2 features such as dynamic memory allocation to further increase the efficiency, scalability and utility of our approach.

Finally, to enhance our overall goals of this dissertation — enhancing the use and understanding of privacy notions and techniques — the following avenues may be promising: study of generic privacy, making privacy notions more accessible to non-experts such as software developers, and evangelising the general public to make privacy a default, not merely a feature that has to be implemented to comply with governance.

7.5 Personal Conclusions

Looking back over the last four years, I still feel a sense of joy and excitement. While this dissertation can be seen as the result of a long and hard journey, it has taught me personally a lot of invaluable life skills (*e.g.* sturdiness, challenge assumptions/-facts/approaches), career skills (*e.g.* independent working style), academic skills (*e.g.* research, out-of-box thinking, technical report writing), soft-skills (*e.g.* presentation skills, openness to other approaches/opinions) and various other real-world skills.

A productive day, for me, is a day when I learn something new. Overall, I had many such days during my endeavor over the last four years. Coming from a different, yet, related field of cryptographic research to privacy research, my learning curve has been immense. During my undergrad and masters I've always been told that for in-depth knowledge it 'requires' an expert with the specific field under question, thus, I set my goal to become such an expert within the context of this dissertation.

This has been proven to be easier said than done, especially if you are lacking knowledge in the field of research, research skills and a clear road map of getting to your goal. As a direct consequence, a certain sturdiness and persistence is required from tenacious researchers to tackle such challenges.

In this context, in my first year I followed in the footsteps of my past experiences — I set out to explore differences between the existing literature of secure multi-party schemes and to explore novel opportunities enabled by the SGX while studying use cases with those privacy implications. In my second and third year, I re-focused on re-lease privacy studying privacy notions, coming up with and exploring privacy games. Halfway through my third year I started writing on what I often considered as a never ending array of words. To my surprise it turned out to be final.

What I've learned throughout this journey is hard to summarise in a few sentences. However, I want to try to state a few bullet points that I believe are valuable insights for anyone (or, at least, what I've taken from my experiences):

(a) **Proper research requires time.** While it is hard to predict the challenges and research angles that will turn out fruitful when tackling it, it often takes much more time than initially assumed. In other words, while exploring a topic, one learns a lot about side effects, convolutions and constraints of a stated problem.

(b) **'Everything' is research (aka there's no bad research).** You might be lazy (that's another story), but by failing in trying to achieve one's goals, one often learns

more than when succeeding on the first try. Keep being interested and trying to tackle a problem from various 'unconventional'[7] ways leads to alternative solutions.

(c) **Keep lean and test.** There is a breaking point of reading existing literature and starting to try something new. Every so often, I ended up reading seemingly endless amount of literature only to realise that it is on me to come up with a new ideas to solve a problem which hasn't been solved before. Nobody is doing the research for you. It takes guts to leap over this step and start doing something 'novel'.

(d) **Divide and conquer.** Tackling a too 'big' problem from the start of your research often confuses one and leads to nowhere. Research typically takes the approach to divide a big problem into small parts of individual tasks (and, thereof, small improvements) which, when combined in a novel way, may lead to stark improvement. From a different viewpoint, it's the little things which end up changing the current way of thinking.

(e) **You know nothing.** Realising that one knows/understands nothing is often the best starting point. It promotes the researcher to close this gap and fill this 'emptiness' with potentially novel knowledge. Research is ever expanding — there will always be a new idea, a new experiment, a next paper. Understanding where the researcher is placed in this maze simplifies the 'hero's' road from the 'abyss' to the 'return' [92].

[7] In this context, it requires the researcher to think-out-of-the-box and be creative to explore non-obvious ways to approach a given problem.

BIBLIOGRAPHY

[1] Health insurance portability and accountability act of 1996. http://legislink.org/us/pl-104-191 (1996)

[2] Information technology — security techniques — privacy framework. Standard ISO/IEC 29100:2011, International Organization for Standardization, Geneva, CH (Dec 2011)

[3] Information technology – security techniques – anonymous digital signatures – part 1: General. Standard ISO/IEC 20008-1:2013, International Organization for Standardization, Geneva, CH (Dec 2013)

[4] Information technology – security techniques – anonymous entity authentication – part 1: General. Standard ISO/IEC 20009-1:2013, International Organization for Standardization, Geneva, CH (Aug 2013)

[5] Information technology — trusted platform module library — part 1: Architecture. Standard ISO/IEC 11889-1:2015, International Organization for Standardization, Geneva, CH (Aug 2015)

[6] Student numbers. https://www.ox.ac.uk/about/facts-and-figures/student-numbers (2017)

[7] Information technology — security techniques — privacy architecture framework. Standard ISO/IEC 29101:2018, International Organization for Standardization, Geneva, CH (Nov 2018)

[8] Information technology — security techniques — privacy framework — amendment 1: Clarifications. Standard ISO/IEC 29100:2011/Amd 1:2018, International Organization for Standardization, Geneva, CH (Jun 2018)

[9] Privacy enhancing data de-identification terminology and classification of techniques. Standard ISO/IEC 20889:2018, International Organization for Standardization, Geneva, CH (Nov 2018)

[10] Abadi, M., Chu, A., Goodfellow, I., McMahan, H.B., Mironov, I., Talwar, K., Zhang, L.: Deep learning with differential privacy. pp. 308–318. CCS '16, ACM (2016)

[11] Abidin, A., Aly, A., Cleemput, S., Mustafa, M.A.: Secure and privacy-friendly local electricity trading and billing in smart grid. CoRR abs/1801.08354 (2018), http://arxiv.org/abs/1801.08354

[12] Abou-Nasr, M., Lessmann, S., Stahlbock, R., Weiss, G.M.: Real World Data Mining Applications. Springer (2014)

[13] Acar, A., Berkay Celik, Z., Aksu, H., Selcuk Uluagac, A., McDaniel, P.: Achieving Secure and Differentially Private Computations in Multiparty Settings. arXiv e-prints (Jul 2017)

[14] Acar, A., Aksu, H., Uluagac, A.S., Conti, M.: A survey on homomorphic encryption schemes: Theory and implementation. ACM Computing Surveys (CSUR) 51(4), 1–35 (2018)

[15] Achenbach, D., Huber, M., Müller-Quade, J., Rill, J.: Closing the Gap: A Universal Privacy Framework for Outsourced Data, pp. 134–151. Springer International Publishing, Cham (2016)

[16] Acquisti, A., John, L.K., Loewenstein, G.: What is privacy worth? The Journal of Legal Studies 42(2), 249–274 (2013)

[17] Acquisti, A., Taylor, C., Wagman, L.: The economics of privacy. Journal of Economic Literature 54(2), 442–92 (June 2016), https://www.aeaweb.org/articles?id=10.1257/jel.54.2.442

[18] Adida, B.: Helios: Web-based open-audit voting. In: Proceedings of the 17th USENIX Security Symposium, July 28-August 1, 2008, San Jose, CA, USA. pp. 335–348 (2008), http://www.usenix.org/events/sec08/tech/full_papers/adida/adida.pdf

[19] Aggarwal, C., Yu, P.: A General Survey of Privacy-Preserving Data Mining Models and Algorithms. In: Aggarwal, C., Yu, P. (eds.) Privacy-Preserving Data Mining: Models and Algorithms. pp. 11–52. Springer (2008)

[20] Aggarwal, C.C.: Data Mining: The Textbook. Springer Publishing Company, Incorporated (2015)

[21] Aggarwal, C.C., Yu, P.S.: Privacy-Preserving Data Mining: A Survey, pp. 431–460. Springer US, Boston, MA (2008), https://doi.org/10.1007/978-0-387-48533-1_18

[22] Aggarwal, C.C., Yu, P.S.: Privacy-Preserving Data Mining: Models and Algorithms. Springer Publishing Company, Incorporated, 1 edn. (2008)

[23] Agrawal, D., Aggarwal, C.C.: On the design and quantification of privacy preserving data mining algorithms. In: Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. pp. 247–255. PODS '01, ACM, New York, NY, USA (2001), http://doi.acm.org/10.1145/375551.375602

[24] Agrawal, R., Srikant, R.: Privacy-Preserving Data Mining. SIGMOD Rec. 29(2), 439–450 (2000)

[25] Alaggan, M., Gambs, S., Kermarrec, A.M.: Heterogeneous Differential Privacy. ArXiv e-prints (Apr 2015)

[26] Albrecht, M., Chase, M., Chen, H., Ding, J., Goldwasser, S., Gorbunov, S., Halevi, S., Hoffstein, J., Laine, K., Lauter, K., Lokam, S., Micciancio, D., Moody, D., Morrison, T., Sahai, A., Vaikuntanathan, V.: Homomorphic encryption security standard. Tech. rep., HomomorphicEncryption.org, Toronto, Canada (November 2018)

[27] Albrecht, M.R., Rechberger, C., Schneider, T., Tiessen, T., Zohner, M.: Ciphers for mpc and fhe. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology – EUROCRYPT 2015. pp. 430–454. Springer Berlin Heidelberg, Berlin, Heidelberg (2015)

[28] Alwen, J., Hirt, M., Maurer, U., Patra, A., Raykov, P.: Anonymous authentication with shared secrets. In: Aranha, D.F., Menezes, A. (eds.) Progress in Cryptology - LATINCRYPT 2014. pp. 219–236. Springer International Publishing, Cham (2015)

[29] Anati, I., Gueron, S. Johnson, S., Scarlata, V.: Innovative Technology for CPU based Attestation and Sealing. In: Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy. HASP 2013, vol. 13. ACM (2013)

[30] Andersson, C., Lundin, R.: On the Fundamentals of Anonymity Metrics (2008)

[31] Ankele, R., Simpson, A.: On the performance of a trustworthy remote entity in comparison to secure multi-party computation. In: 2017 IEEE Trustcom/BigDataSE/ICESS. pp. 1115–1122 (Aug 2017)

[32] Ankele, R., Simpson, A.: Analysis and evaluation of syntactic privacy notions and games. In: 2018 16th Annual Conference on Privacy, Security and Trust (PST). pp. 1–2 (Aug 2018)

[33] Ankele, R.: On privacy-preserving data mining using trustworthy remote entities: Case study. Tech. rep., University of Oxford (2015)

[34] Ankele, R., Küçük, K.A., Martin, A., Simpson, A., Paverd, A.: Applying the trustworthy remote entity to privacy-preserving multiparty computation: Requirements and criteria for large-scale applications. In: 2016 Intl IEEE Conferences on Ubiquitous Intelligence Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld). pp. 414–422. IEEE (July 2016)

[35] Ankele, R., Simpson, A.: Abstracting syntactic privacy notions via privacy games. In: The 16th IEEE International Conference on Advanced and Trusted Computing 2019 (ATC 2019). IEEE, Leicester, United Kingdom (Great Britain) (Aug 2019)

[36] Ankele, R., Simpson, A.: Analysing and evaluating syntactic privacy games via a recommender systems case study. In: The 16th IEEE International Conference on Advanced and Trusted Computing 2019 (ATC 2019). IEEE, Leicester, United Kingdom (Great Britain) (Aug 2019)

[37] APEC: Apec privacy framework. https://www.apec.org/-/media/APEC/Publications/2017/8/APEC-Privacy-Framework-(2015)/217_ECSG_2015-APEC-Privacy-Framework.pdf (2017)

[38] Archer, D.W., Bogdanov, D., Pinkas, B., Pullonen, P.: Maturity and performance of programmable secure computation. IEEE Security Privacy 14(5), 48–56 (Sep 2016)

[39] Archer, D.W., Bogdanov, D., Lindell, Y., Kamm, L., Nielsen, K., Pagter, J.I., Smart, N.P., Wright, R.N.: From Keys to Databases—Real-World Applications of Secure Multi-Party Computation. The Computer Journal 61(12), 1749–1771 (09 2018)

[40] Arnautov, S., Trach, B., Gregor, F., Knauth, T., Martin, A., Priebe, C., Lind, J., Muthukumaran, D., O'Keeffe, D., Stillwell, M.L., Goltzsche, D., Eyers, D., Kapitza, R., Pietzuch, P., Fetzer, C.: Scone: Secure linux containers with intel sgx. In: Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation. pp. 689–703. OSDI'16, USENIX Association, Berkeley, CA, USA (2016), http://dl.acm.org/citation.cfm?id=3026877.3026930

[41] Asghar, M.R., Dán, G., Miorandi, D., Chlamtac, I.: Smart meter data privacy: A survey. IEEE Communications Surveys Tutorials 19(4), 2820–2835 (Fourthquarter 2017)

[42] Association, E.N.: ENA Functional Requirements for Electricity Smart Meters. Tech. Rep. v2.0, Energy Networks Association (2010)

[43] Aumann, Y., Lindell, Y.: Security against covert adversaries: Efficient protocols for realistic adversaries. In: Vadhan, S.P. (ed.) Theory of Cryptography. pp. 137–156. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)

[44] Aumasson, J., Merino, L.: SGX Secure Enclaves in Practice: Security and Crypto Review. https://www.blackhat.com/docs/us-16/materials/us-16-Aumasson-SGX-Secure-Enclaves-In-Practice-Security-And-Crypto-Review-wp.pdf (2016)

[45] Backstrom, L., Dwork, C., Kleinberg, J.: Wherefore art thou r3579x?: Anonymized social networks, hidden patterns, and structural steganography. Commun. ACM 54(12), 133–141 (Dec 2011), http://doi.acm.org/10.1145/2043174.2043199

[46] Badsha, S., Yi, X., Khalil, I.: A practical privacy-preserving recommender system. Data Science and Engineering 1(3), 161–177 (Sep 2016), https://doi.org/10.1007/s41019-016-0020-2

[47] Bahmani, R., Barbosa, M., Brasser, F., Portela, B., Sadeghi, A.R., Scerri, G., Warinschi, B.: Secure multiparty computation from sgx. In: Kiayias, A. (ed.) Financial Cryptography and Data Security. pp. 477–497. Springer International Publishing, Cham (2017)

[48] Balasch, J., Rial, A., Troncoso, C., Preneel, B., Verbauwhede, I., Geuens, C.: Pretp: Privacy-preserving electronic toll pricing. pp. 63–78. USENIX Association (11 2010)

[49] Barbaro, M., Zeller, T.: A face is exposed for aol searcher no. 4417749. https://www.nytimes.com/2006/08/09/technology/09aol.html (2006)

[50] Barker, E.B., Roginsky, A.L.: Sp 800-131a. transitions: Recommendation for transitioning the use of cryptographic algorithms and key lengths. Tech. rep., Gaithersburg, MD, United States (2011)

[51] Beaver, D., Micali, S., Rogaway, P.: The Round Complexity of Secure Protocols. In: Proceedings of the 22nd Annual ACM Symposium on Theory of Computing. pp. 503–513. STOC 1990, ACM (1990)

[52] Beaver, D.: Efficient multiparty protocols using circuit randomization. In: Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology. pp. 420–432. CRYPTO '91, Springer-Verlag, London, UK, UK (1992), http://dl.acm.org/citation.cfm?id=646756.705383

[53] Beerliová-Trubíniová, Z., Hirt, M.: Perfectly-secure mpc with linear communication complexity. In: Proceedings of the 5th Conference on Theory of Cryptography. pp. 213–230. TCC'08, Springer-Verlag, Berlin, Heidelberg (2008), http://dl.acm.org/citation.cfm?id=1802614.1802632

[54] Bellare, M., Namprempre, C.: Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In: Okamoto, T. (ed.) Advances in Cryptology — ASIACRYPT 2000. pp. 531–545. Springer Berlin Heidelberg, Berlin, Heidelberg (2000)

[55] Bellare, M., Rogaway, P.: The game-playing technique. Cryptology ePrint Archive, Report 2004/331 (2004), http://eprint.iacr.org/2004/331, version 20050712:214115

[56] Ben-David, A., Nisan, N., Pinkas, B.: FairplayMP: A System for Secure Multiparty Computation. In: Proceedings of the 15th ACM Conference on Computer and Communications Security. pp. 257–266. CCS 2008, ACM (2008)

[57] Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness Theorems for Non-cryptographic Fault-tolerant Distributed Computation. In: Proceedings of the 20th Annual ACM Symposium on Theory of Computing. pp. 1–10. STOC 1988, ACM (1988)

[58] Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046 (2018), https://eprint.iacr.org/2018/046

[59] Benger, N., Bernhard, D., Catalano, D., Charlemagne, M., Conti, D., Cubaleska, B., Fernando, H., Fiore, D., Galbraith, S., Galindo, D., Hermans, J., Iovino, V., Jager, T., Kohlweiss, M., Libert, B., Lindner, R., Loehr, H., Lynch, D., Moloney, R., Ouafi, K., Pinkas, B., Polach, F., Di Raimondo, M., Rückert, M., Schneider, M., Singh, V., Smart, N., Stam, M., Vercauteren, F., Villar Santos, J., Williams, S.: Final report on main computational assumptions incryptography. Tech. Rep. D.MAYA.6 (2013)

[60] Bertino, E., Fovino, I.N., Provenza, L.P.: A framework for evaluating privacy preserving data mining algorithms*. Data Min. Knowl. Discov. 11(2), 121–154 (Sep 2005), http://dx.doi.org/10.1007/s10618-005-0006-6

[61] Binu, V.P., Nair, D.G., Sreekumar, A.: Secret Sharing Homomorphism and Secure E-voting. arXiv e-prints arXiv:1602.05372 (Feb 2016)

[62] Bizzozero, F., Gruosso, G., Vezzini, N.: A time-of-use-based residential electricity demand model for smart grid applications. In: 2016 IEEE 16th International Conference on Environment and Electrical Engineering (EEEIC). pp. 1–6 (June 2016)

[63] Blakley, G.R.: Safeguarding cryptographic keys. In: Managing Requirements Knowledge, International Workshop on. p. 313. IEEE Computer Society, Los Alamitos, CA, USA (jun 1979), https://doi.ieeecomputersociety.org/10.1109/AFIPS.1979.98

[64] Blum, A., Dwork, C., McSherry, F., Nissim, K.: Practical privacy: The sulq framework. In: Proceedings of the Twenty-fourth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. pp. 128–138. PODS '05, ACM, New York, NY, USA (2005), http://doi.acm.org/10.1145/1065167.1065184

[65] Blum, A., Ligett, K., Roth, A.: A learning theory approach to non-interactive database privacy. In: Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing. pp. 609–618. STOC '08, ACM, New York, NY, USA (2008), http://doi.acm.org/10.1145/1374376.1374464

[66] Blum, M.: Coin flipping by telephone a protocol for solving impossible problems. SIGACT News 15(1), 23–27 (Jan 1983), http://doi.acm.org/10.1145/1008908.1008911

[67] Bogdanov, D., Laur, S., Willemson, J.: Sharemind: A Framework for Fast Privacy-Preserving Computations. In: Jajodia, S., Lopez, J. (eds.) Proceedings of the 13th European Symposium on Research in Computer Security - ESORICS 2008. Lecture Notes in Computer Science, vol. 5283, pp. 192–206. Springer (2008)

[68] Bogdanov, D., Kamm, L., Kubo, B., Rebane, R., Sokk, V., Talviste, R.: Students and taxes: a privacy-preserving study using secure computation. Proceedings on Privacy Enhancing Technologies 2016(3), 117–135 (2016), https://content.sciendo.com/view/journals/popets/2016/3/article-p117.xml

[69] Bogdanov, D., Niitsoo, M., Toft, T., Willemson, J.: High-performance secure multi-party computation for data mining applications. International Journal of Information Security 11(6), 403–418 (Nov 2012), https://doi.org/10.1007/s10207-012-0177-2

[70] Bogetoft, P., Christensen, D.L., Damgård, I., Geisler, M., Jakobsen, T., Krøigaard, M., Nielsen, J.D., Nielsen, J.B., Nielsen, K., Pagter, J., Schwartzbach, M., Toft, T.: Secure multiparty computation goes live. In: Dingledine, R., Golle, P. (eds.) Financial Cryptography and Data Security. pp. 325–343. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)

[71] Bogetoft, P., Damgård, I., Jakobsen, T., Nielsen, K., Pagter, J., Toft, T.: A practical implementation of secure auctions based on multiparty integer computation. In: Di Crescenzo, G., Rubin, A. (eds.) Financial Cryptography and Data Security. pp. 142–147. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)

[72] Bohli, J.M., Pashalidis, A.: Relations Among Privacy Notions. ACM Trans. Inf. Syst. Secur. 14(1) (2011)

[73] Boyle, E., Chung, K., Pass, R.: Large-Scale Secure Computation: Multi-party Computation for (Parallel) RAM Programs. In: Gennaro, R., Robshaw, M. (eds.) Proc. of the 35th Cryptology Conf. - CRYPTO 2015. Lecture Notes in C.S., vol. 9216, pp. 742–762. Springer (2015)

[74] Braeken, A., Kumar, P., Martin, A.: Efficient and privacy-preserving data aggregation and dynamic billing in smart grid metering networks. Energies 11, 2085 (08 2018)

[75] Brakerski, Z.: Fundamentals of fully homomorphic encryption (10 2019)

[76] Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. In: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference. pp. 309–325. ITCS '12, ACM, New York, NY, USA (2012), http://doi.acm.org/10.1145/2090236.2090262

[77] Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-lwe and security for key dependent messages. In: Proceedings of the 31st Annual Conference on Advances in Cryptology. pp. 505–524. CRYPTO'11, Springer-Verlag, Berlin, Heidelberg (2011), http://dl.acm.org/citation.cfm?id=2033036.2033075

[78] Brandenburger, M., Cachin, C., Lorenz, M., Kapitza, R.: Rollback and forking detection for trusted execution environments using lightweight collective memory. In: 2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). pp. 157–168 (June 2017)

[79] Brasser, F., Müller, U., Dmitrienko, A., Kostiainen, K., Capkun, S., Sadeghi, A.R.: Software grand exposure: SGX cache attacks are practical. In: 11th USENIX Workshop on Offensive Technologies (WOOT 17). USENIX Association, Vancouver, BC (2017), https://www.usenix.org/conference/woot17/workshop-program/presentation/brasser

[80] Bresson, E., Catalano, D., Fazio, N., Nicolosi, A., Yung, M.: Output Privacy in Secure Multiparty Computation. In: Proceedings of the 3rd Yet Another Conference on Cryptography (2006)

[81] Brickell, E., Camenisch, J., Chen, L.: Direct Anonymous Attestation. In: Proceedings of the 11th ACM Conference on Computer and Communications Security. pp. 132–145. CCS 2004, ACM (2004)

[82] Brickell, E., Li, J.: Enhanced privacy id from bilinear pairing for hardware authentication and attestation. In: 2010 IEEE Second International Conference on Social Computing. pp. 768–775 (Aug 2010)

[83] Brickell, E., Chen, L., Li, J.: Simplified security notions of direct anonymous attestation and a concrete scheme from pairings. International Journal of Information Security 8(5), 315–330 (Oct 2009), https://doi.org/10.1007/s10207-009-0076-3

[84] Brickell, E., Li, J.: Enhanced privacy id: A direct anonymous attestation scheme with enhanced revocation capabilities. In: Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society. pp. 21–30. WPES '07, ACM, New York, NY, USA (2007), http://doi.acm.org/10.1145/1314333.1314337

[85] Brickell, E., Li, J.: Enhanced privacy id from bilinear pairing. IACR Cryptology ePrint Archive 2009, 95 (01 2009)

[86] Brown, I.: Britain's smart meter programme: A case study in privacy by design. Int. Rev. Law Comput. Technol. 28(2), 172–184 (May 2014), http://dx.doi.org/10.1080/13600869.2013.801580

[87] Bulck, J.V., Minkin, M., Weisse, O., Genkin, D., Kasikci, B., Piessens, F., Silberstein, M., Wenisch, T.F., Yarom, Y., Strackx, R.: Foreshadow: Extracting the keys to the intel SGX kingdom with transient out-of-order execution. In: 27th USENIX Security Symposium (USENIX Security 18). p. 991–1008. USENIX Association, Baltimore, MD (2018), https://www.usenix.org/conference/usenixsecurity18/presentation/bulck

[88] Bun, M., Steinke, T.: Concentrated differential privacy: Simplifications, extensions, and lower bounds. In: Proceedings, Part I, of the 14th International Conference on Theory of Cryptography - Volume 9985. pp. 635–658. Springer-Verlag New York, Inc., New York, NY, USA (2016), https://doi.org/10.1007/978-3-662-53641-4_24

[89] Burkhart, M., Strasser, M., Many, D., Dimitropoulos, X.: SEPIA: Privacy-preserving Aggregation of Multi-domain Network Events and Statistics. In: Proceedings of the 19th USENIX Conference on Security. pp. 15–15. SEC 2010, USENIX Association (2010)

[90] Calandrino, J.A., Kilzer, A., Narayanan, A., Felten, E.W., Shmatikov, V.: "you might also like:" privacy risks of collaborative filtering. In: 2011 IEEE Symposium on Security and Privacy. pp. 231–246 (May 2011)

[91] Calmon, F.P., Makhdoumi, A., Médard, M.: Fundamental limits of perfect privacy. In: 2015 IEEE International Symposium on Information Theory (ISIT). pp. 1796–1800 (June 2015)

[92] Campbell, J.: The Hero with a Thousand Faces. Pantheon Books (1949)

[93] Canetti, R.: Universally Composable Security: A New Paradigm for Cryptographic Protocols. In: Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science. pp. 136–145. FOCS 2001, IEEE Computer Society (2001)

[94] Canetti, R.: Studies in Secure Multiparty Computation and Applications. Ph.D. thesis, The Weizmann Institute of Science (1995)

[95] Canetti, R.: Security and composition of multiparty cryptographic protocols. J. Cryptology 13(1), 143–202 (2000), https://doi.org/10.1007/s001459910006

[96] Canetti, R., Krawczyk, H.: Universally composable notions of key exchange and secure channels. In: Knudsen, L.R. (ed.) Advances in Cryptology — EUROCRYPT 2002. pp. 337–351. Springer Berlin Heidelberg, Berlin, Heidelberg (2002)

[97] Cao, J., Karras, P.: Publishing microdata with a robust privacy guarantee. Proc. VLDB Endow. 5(11), 1388–1399 (Jul 2012), http://dx.doi.org/10.14778/2350229.2350255

[98] Cao, J., Karras, P., Raïssi, C., Tan, K.L.: ρuncertaintyy: Inference-proof transaction anonymization. Proc. VLDB Endow. 3(1-2), 1033–1044 (Sep 2010), http://dx.doi.org/10.14778/1920841.1920971

[99] Cavoukian, A., Kursawe, K.: Implementing privacy by design: The smart meter case. In: 2012 International Conference on Smart Grid (SGE). pp. 1–8 (Aug 2012)

[100] Cavoukian, A.: Privacy by Design - The 7 Foundational Principles. https://www.ipc.on.ca/images/Resources/7foundationalprinciples.pdf (2009)

[101] of Certified Public Accountants, A.I., of Chartered Accountants, C.I.: Generally accepted privacy principles. Tech. rep. (2009)

[102] Chaum, D.: Blind signatures for untraceable payments. In: Chaum, D., Rivest, R.L., Sherman, A.T. (eds.) Advances in Cryptology. pp. 199–203. Springer US, Boston, MA (1983)

[103] Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM 24(2), 84–90 (Feb 1981), http://doi.acm.org/10.1145/358549.358563

[104] Chawla, S., Dwork, C., McSherry, F., Talwar, K.: On privacy-preserving histograms. In: Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence. pp. 120–127. UAI'05, AUAI Press, Arlington, Virginia, United States (2005), http://dl.acm.org/citation.cfm?id=3020336.3020351

[105] Checkoway, S., Shacham, H.: Iago attacks: Why the system call api is a bad untrusted rpc interface. In: Proceedings of the Eighteenth International Conference on Architectural Support for Programming Languages and Operating Systems. pp. 253–264. ASPLOS '13, ACM, New York, NY, USA (2013), http://doi.acm.org/10.1145/2451116.2451145

[106] Chen, G., Chen, S., Xiao, Y., Zhang, Y., Lin, Z., Lai, T.H.: Sgxpectre attacks: Leaking enclave secrets via speculative execution. CoRR abs/1802.09085 (2018), http://arxiv.org/abs/1802.09085

[107] Chen, L., Li, J.: A note on the chen–morrissey–smart daa scheme. Information Processing Letters 110(12), 485 – 488 (2010), http://www.sciencedirect.com/science/article/pii/S0020019010000980

[108] Chen, L., Morrissey, P., Smart, N.P.: On proofs of security for daa schemes. In: Baek, J., Bao, F., Chen, K., Lai, X. (eds.) Provable Security. pp. 156–175. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)

[109] Chen, L., Morrissey, P., Smart, N.P.: Pairings in trusted computing. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing-Based Cryptography – Pairing 2008. pp. 1–17. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)

[110] Chen, L., Page, D., P. Smart, N.: On the design and implementation of an efficient daa scheme. pp. 223–237 (11 2010)

[111] Chen, M.S., Han, J., Yu, P.S.: Data mining: An overview from a database perspective. IEEE Trans. on Knowl. and Data Eng. 8(6), 866–883 (Dec 1996)

[112] Chen, Y., Sheffet, O., Vadhan, S.: Privacy games. In: 10th Conference on Web and Internet Economics (WINE). Beijing, China (December 2014)

[113] Cho, H., Wu, D.J., Berger, B.: Secure genome-wide association analysis using multiparty computation. Nature Biotechnology 36 (2018)

[114] Choi, J., Butler, K.: Secure multiparty computation and trusted hardware: Examining adoption challenges and opportunities. Security and Communication Networks 2019(1368905) (2019)

[115] Clarke, R.: Introduction to dataveillance and information privacy, and definitions of terms. http://www.rogerclarke.com/DV/Intro.html (1997)

[116] Clauß, S., Schiffner, S.: Structuring Anonymity Metrics. In: Proceedings of the 2nd ACM Workshop on Digital Identity Management. pp. 55–62. DIM '06, ACM (2006)

[117] Clifton, C., Kantarcioglu, M., Vaidya, J.: Defining Privacy for Data Mining. In: Proccedings of the National Science Foundation Workshop on Next Generation Data Mining. pp. 126–133 (2002)

[118] Clifton, C., Kantarcioglu, M., Vaidya, J., Lin, X., Zhu, M.: Tools for privacy preserving distributed data mining. SIGKDD Explor. Newsl. 4(2), 28–34 (2002)

[119] Clifton, C., Tassa, T.: On syntactic anonymity and differential privacy. Trans. Data Privacy 6(2), 161–183 (Aug 2013), http://dl.acm.org/citation.cfm?id=2612167.2612170

[120] Coker, G., Guttman, J., Loscocco, P., Herzog, A., Millen, J., O'Hanlon, B., Ramsdell, J., Segall, A., Sheehy, J., Sniffen, B.: Principles of remote attestation. International Journal of Information Security 10(2), 63–81 (Jun 2011), https://doi.org/10.1007/s10207-011-0124-7

[121] Coron, J., Mandal, A., Naccache, D., Tibouchi, M.: Fully Homomorphic Encryption over the Integers with Shorter Public Keys. In: Rogaway, P. (ed.) Proc. of the 31st Cryptology Conf. - CRYPTO 2011. Lecture Notes in C.S., vol. 6841, pp. 487–504. Springer (2011)

[122] Corporation, I.: Intel Software Guard Extensions Programming Reference. https://software.intel.com/sites/default/files/managed/48/88/329298-002.pdf (2014)

[123] Corporation, I.: Enclave writer's guide. https://software.intel.com/sites/default/files/managed/ae/48/Software-Guard-Extensions-Enclave-Writers-Guide.pdf (2015)

[124] Corporation, I.: A cost-effective foundation for end-to-end iot security. https://intel-epid-sdk.github.io/download/intel-epid-white-paper.pdf (2016)

[125] Corporation, I.: Intel software guard extensions (intel sgx) - developer guide. https://download.01.org/intel-sgx/linux-2.4/docs/Intel_SGX_Developer_Guide.pdf (2018)

[126] Corporation, I.: Intel software guard extensions (intel sgx) data center attestation primitives: Ecdsa quote library api. https://download.01.org/intel-sgx/dcap-1.0.1/docs/Intel_SGX_ECDSA_QuoteGenReference_DCAP_API_Linux_1.0.1.pdf (2018)

[127] Corporation, I.: Intel software guard extensions (intel sgx) sdk for linux os. https://download.01.org/intel-sgx/linux-2.4/docs/Intel_SGX_Developer_Reference_Linux_2.4_Open_Source.pdf (2018)

[128] Costan, V., Devadas, S.: Intel sgx explained. Cryptology ePrint Archive, Report 2016/086 (2016), https://eprint.iacr.org/2016/086

[129] Cramer, R., Damgård, I., Maurer, U.: General Secure Multi-party Computation from any Linear Secret-Sharing Scheme. In: Preneel, B. (ed.) Proccedings of the International Conference on the Theory and Application of Cryptographic Techniques - EUROCRYPT 2000. Lecture Notes in Computer Science, vol. 1807, pp. 316–334. Springer (2000)

[130] Cramer, R., Damgård, I., Nielsen, J.: Multiparty Computation from Threshold Homomorphic Encryption. In: Pfitzmann, B. (ed.) Proceeding of the International Conference on the Theory and Application of Cryptographic Techniques - EUROCRYPT 2001. Lecture Notes in Computer Science, vol. 2045, pp. 280–300. Springer (2001)

[131] Cramer, R., Damgård, I., Ishai, Y.: Share conversion, pseudorandom secret-sharing and applications to secure computation. In: Kilian, J. (ed.) Theory of Cryptography. pp. 342–362. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)

[132] Cuff, P., Yu, L.: Differential privacy as a mutual information constraint. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 43–54. CCS '16, ACM, New York, NY, USA (2016), http://doi.acm.org/10.1145/2976749.2978308

[133] D Smith, W.: Cryptography meets voting. https://rangevoting.org/WarrenSmithPages/homepage/cryptovot.pdf (10 2005)

[134] Dalenius, T.: Towards a methodology for statistical disclosure control. Statistik Tidskrift 15, 2–1 (1977)

[135] Damgård, I., Geisler, M., Krøigaard, M., Nielsen, J.: Asynchronous Multiparty Computation: Theory and Implementation. In: Jarecki, S., Tsudik, G. (eds.) Proceedings of the 12th International Conference on Practice and Theory in Public Key Cryptography - PKC 2009. Lecture Notes in Computer Science, vol. 5443, pp. 160–179. Springer (2009)

[136] Damgård, I., Pastro, V., Smart, N., Zakarias, S.: Multiparty Computation from Somewhat Homomorphic Encryption. In: Safavi-Naini, R., Canetti, R. (eds.) Proceedings of the 32nd Annual Cryptology Conference - CRYPTO 2012. Lecture Notes in Computer Science, vol. 7417, pp. 643–662. Springer (2012)

[137] Damgård, I., Damgård, K., Nielsen, K., Nordholt, P.S., Toft, T.: Confidential benchmarking based on multiparty computation. In: Grossklags, J., Preneel, B. (eds.) Financial Cryptography and Data Security. pp. 169–187. Springer Berlin Heidelberg, Berlin, Heidelberg (2017)

[138] Damgård, I., Geisler, M., Krøigaard, M.: Efficient and secure comparison for on-line auctions. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) Information Security and Privacy. pp. 416–430. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)

[139] Damgård, I., Ishai, Y., Krøigaard, M.: Perfectly secure multiparty computation and the computational overhead of cryptography. In: Gilbert, H. (ed.) Advances in Cryptology – EUROCRYPT 2010. pp. 445–465. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)

[140] Damgård, I., Keller, M., Larraia, E., Miles, C., Smart, N.P.: Implementing aes via an actively/covertly secure dishonest-majority mpc protocol. In: Visconti, I., De Prisco, R. (eds.) Security and Cryptography for Networks. pp. 241–263. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)

[141] Damgård, I., Keller, M., Larraia, E., Pastro, V., Scholl, P., Smart, N.P.: Practical covertly secure mpc for dishonest majority – or: Breaking the spdz limits. In: Crampton, J., Jajodia, S., Mayes, K. (eds.) Computer Security – ESORICS 2013. pp. 1–18. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)

[142] Damgård, I., Nielsen, J.B.: Scalable and unconditionally secure multiparty computation. In: Menezes, A. (ed.) Advances in Cryptology - CRYPTO 2007. pp. 572–590. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)

[143] Demmler, D., Schneider, T., Zohner, M.: ABY – A Framework for Efficient Mixed-Protocol Secure Two-Party Computation. NDSS 2015, The Internet Society (2015)

[144] Denning, D.E.: Secure statistical databases with random sample queries. ACM Trans. Database Syst. 5(3), 291–315 (Sep 1980), https://doi.org/10.1145/320613.320616

[145] Denning, D.E., Denning, P.J.: Data security. ACM Comput. Surv. 11(3), 227–249 (Sep 1979), https://doi.org/10.1145/356778.356782

[146] Denning, D.E., Denning, P.J., Schwartz, M.D.: The tracker: A threat to statistical database security. ACM Trans. Database Syst. 4(1), 76–96 (Mar 1979), https://doi.org/10.1145/320064.320069

[147] Diaz, C., Claessens, J., Seys, S., Preneel, B.: Information Theory and Anonymity. In: Macq, B., Quisquater, J.J. (eds.) Proceedings of the 23rd Symposium on Information Theory in the Benelux. pp. 179–186. Werkgemeenschap voor Informatie- en Communicatietheorie (2002)

[148] Diaz, C., Claessens, J., Seys, S., Preneel, B.: Towards Measuring Anonymity (2002)

[149] Diaz, M., Sankar, L., Kairouz, P.: On the contractivity of privacy mechanisms (2018)

[150] Dimitrakakis, C., Nelson, B., Zhang, Z., Mitrokotsa, A., Rubinstein, B.I.P.: Differential privacy for bayesian inference through posterior sampling. J. Mach. Learn. Res. 18(1), 343–381 (Jan 2017), http://dl.acm.org/citation.cfm?id=3122009.3122020

[151] Dinur, I., Nissim, K.: Revealing information while preserving privacy. In: Proceedings of the Twenty-second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. pp. 202–210. PODS '03, ACM, New York, NY, USA (2003), http://doi.acm.org/10.1145/773153.773173

[152] Doganay, M., Pedersen, T., Saygin, Y., Savaş, E., Levi, A.: Distributed Privacy Preserving K-means Clustering with Additive Secret Sharing. In: Proceedings of the International Workshop on Privacy and Anonymity in Information Society. pp. 3–11. PAIS 2008, ACM (2008)

[153] Domingo-Ferrer, J.: Coprivacy: Towards a theory of sustainable privacy. In: Proceedings of the 2010 International Conference on Privacy in Statistical Databases. pp. 258–268. PSD'10, Springer-Verlag, Berlin, Heidelberg (2010)

[154] Doudalis, S., Kotsogiannis, I., Haney, S., Machanavajjhala, A., Mehrotra, S.: One-sided Differential Privacy. ArXiv e-prints (Dec 2017)

[155] Du, W., Atallah, M.J.: Secure multi-party computation problems and their applications: A review and open problems. In: Proceedings of the 2001 Workshop on New Security Paradigms. pp. 13–22. NSPW '01, ACM, New York, NY, USA (2001), http://doi.acm.org/10.1145/508171.508174

[156] Dua, S., Du, X.: Data Mining and Machine Learning in Cybersecurity. Auerbach Publications, Boston, MA, USA, 1st edn. (2011)

[157] Dupree, J.L., Devries, R., Berry, D.M., Lank, E.: Privacy personas: Clustering users via attitudes and behaviors toward security practices. In: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems. p.

5228–5239. CHI '16, Association for Computing Machinery, New York, NY, USA (2016), https://doi.org/10.1145/2858036.2858214

[158] Dwork, C., Rothblum, G.N.: Concentrated Differential Privacy. ArXiv e-prints (Mar 2016)

[159] Dwork, C.: Differential privacy. In: Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II. pp. 1–12 (2006)

[160] Dwork, C.: Differential privacy: A survey of results. In: Agrawal, M., Du, D., Duan, Z., Li, A. (eds.) Theory and Applications of Models of Computation. pp. 1–19. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)

[161] Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., Naor, M.: Our data, ourselves: Privacy via distributed noise generation. In: Vaudenay, S. (ed.) Advances in Cryptology - EUROCRYPT 2006. pp. 486–503. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)

[162] Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: Proceedings of the Third Conference on Theory of Cryptography. pp. 265–284. TCC'06, Springer-Verlag, Berlin, Heidelberg (2006), http://dx.doi.org/10.1007/11681878_14

[163] Dwork, C., Roth, A.: The algorithmic foundations of differential privacy. Found. Trends Theor. Comput. Sci. 9(3–4), 211–407 (Aug 2014), http://dx.doi.org/10.1561/0400000042

[164] Dworkin, M.: Nist special publication 800-38b: Recommendation for block cipher modes of operation: The cmac mode for authentication. Tech. Rep. 800-38B (2016)

[165] Ebadi, H., Sands, D., Schneider, G.: Differential privacy: Now it's getting personal. In: Proceedings of the 42Nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. pp. 69–81. POPL '15, ACM, New York, NY, USA (2015), http://doi.acm.org/10.1145/2676726.2677005

[166] ECHR: European convention on human rights. https://www.echr.coe.int/Documents/Convention_ENG.pdf (1950)

[167] Edman, M., Sivrikaya, F., Yener, B.: A Combinatorial Approach to Measuring Anonymity. In: Intelligence and Security Informatics, 2007 IEEE. pp. 356–363 (2007)

[168] Eigner, F., Kate, A., Maffei, M., Pampaloni, F., Pryvalov, I.: Achieving Optimal Utility for Distributed Differential Privacy Using Secure Multiparty Computation 13, 81 – 105 (2015)

[169] Ejgenberg, Y., Farbstein, M., Levy, M., Lindell, Y.: SCAPI: The Secure Computation Application Programming Interface. Cryptology ePrint Archive, Report 2012/629 (2012)

[170] Ekstrand, M.D., Riedl, J.T., Konstan, J.A.: Collaborative filtering recommender systems. Found. Trends Hum.-Comput. Interact. 4(2), 81–173 (Feb 2011), http://dx.doi.org/10.1561/1100000009

[171] El Makkaoui, K., Ezzati, A., Hssane, A.B.: Challenges of using homomorphic encryption to secure cloud computing. In: 2015 International Conference on Cloud Technologies and Applications (CloudTech). pp. 1–7 (2015)

[172] Elgamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory 31(4), 469–472 (July 1985)

[173] Emam, K.: Guide to the De-Identification of Personal Health Information. Auerbach Publications, New York, USA, 1 edn. (2013)

[174] Enes, V., Silva, P., Almeida, J., Portela, B.: Multi Party Computation - Active Adversaries. https://github.com/vitorenesduarte/spdz/blob/master/report.pdf (2015)

[175] Engelbrecht-Wiggans, R.: Auctions and bidding models: A survey. Management Science 26 (02 1979)

[176] Erlingsson, U., Pihur, V., Korolova, A.: Rappor: Randomized aggregatable privacy-preserving ordinal response. pp. 1054–1067. CCS '14, ACM (2014)

[177] Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). Official Journal of the European Union L119, 1–88 (May 2016), http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=OJ:L:2016:119:TOC

[178] Evans, D., Kolesnikov, V., Rosulek, M.: A Pragmatic Introduction to Secure Multi-Party Computation, vol. 2 (01 2018)

[179] Evfimievski, A., Gehrke, J., Srikant, R.: Limiting privacy breaches in privacy preserving data mining. In: Proceedings of the Twenty-second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. pp. 211–222. PODS '03, ACM, New York, NY, USA (2003), http://doi.acm.org/10.1145/773153.773174

[180] Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P.: Advances in knowledge discovery and data mining. chap. From Data Mining to Knowledge Discovery: An Overview, pp. 1–34. American Association for Artificial Intelligence, Menlo Park, CA, USA (1996)

[181] Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: Cachin, C., Camenisch, J.L. (eds.) Advances in Cryptology - EUROCRYPT 2004. pp. 1–19. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)

[182] Frikken, K.: Privacy-preserving set union. In: Katz, J., Yung, M. (eds.) Applied Cryptography and Network Security. pp. 237–252. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)

[183] Fu, Y., Bauman, E., Quinonez, R., Lin, Z.: Sgx-lapd: Thwarting controlled side channel attacks via enclave verifiable page faults. In: Dacier, M., Bailey, M., Polychronakis, M., Antonakakis, M. (eds.) Research in Attacks, Intrusions, and Defenses. pp. 357–380. Springer International Publishing, Cham (2017)

[184] Fung, B.C.M., Wang, K., Yu, P.S.: Top-down specialization for information and privacy preservation. In: 21st International Conference on Data Engineering (ICDE'05). pp. 205–216 (April 2005)

[185] Fung, B.C.M., Wang, K., Yu, P.S.: Anonymizing classification data for privacy preservation. IEEE Transactions on Knowledge and Data Engineering 19(5), 711–725 (May 2007)

[186] Fung, B.C.M., Cao, M., Desai, B.C., Xu, H.: Privacy protection for rfid data. In: Proceedings of the 2009 ACM Symposium on Applied Computing. pp. 1528–1535. SAC '09, ACM, New York, NY, USA (2009), http://doi.acm.org/10.1145/1529282.1529626

[187] Fung, B.C.M., Wang, K., Chen, R., Yu, P.S.: Privacy-preserving data publishing: A survey of recent developments. ACM Comput. Surv. 42(4), 14:1–14:53 (Jun 2010), http://doi.acm.org/10.1145/1749603.1749605

[188] Ganta, S.R., Kasiviswanathan, S.P., Smith, A.: Composition attacks and auxiliary information in data privacy. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 265–273. KDD '08, ACM (2008), http://doi.acm.org/10.1145/1401890.1401926

[189] Ge, Q., Yarom, Y., Cock, D., Heiser, G.: A survey of microarchitectural timing attacks and countermeasures on contemporary hardware. Journal of Cryptographic Engineering 8(1), 1–27 (Apr 2018), https://doi.org/10.1007/s13389-016-0141-6

[190] Gentry, C., Halevi, S.: Implementing Gentry's Fully-Homomorphic Encryption Scheme. In: Paterson, K. (ed.) Proc. of the 30th Int. Conf. on the Theory and Applications of Cryptographic Techniques - EUROCRYPT 2011. Lecture Notes in C.S., vol. 6632, pp. 129–148. Springer (2011)

[191] Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing. p. 169–178. STOC 09, Association for Computing Machinery, New York, NY, USA (2009), https://doi.org/10.1145/1536414.1536440

[192] Ghosh, A., Roughgarden, T., Sundararajan, M.: Universally utility-maximizing privacy mechanisms. In: Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing. pp. 351–360. STOC '09, ACM, New York, NY, USA (2009), http://doi.acm.org/10.1145/1536414.1536464

[193] Gionis, A., Mazza, A., Tassa, T.: k-anonymization revisited. In: 2008 IEEE 24th International Conference on Data Engineering. pp. 744–753 (April 2008)

[194] Goldreich, O.: Towards a theory of software protection and simulation by oblivious rams. In: Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing. pp. 182–194. STOC '87, ACM, New York, NY, USA (1987), http://doi.acm.org/10.1145/28395.28416

[195] Goldreich, O., Micali, S., Wigderson, A.: How to Play ANY Mental Game. In: Proceedings of the 19th Annual ACM Symposium on Theory of Computing. pp. 218–229. STOC 1987, ACM (1987)

[196] Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems. In: Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing. pp. 291–304. STOC '85, ACM, New York, NY, USA (1985), http://doi.acm.org/10.1145/22145.22178

[197] Goldwasser, S., Tauman Kalai, Y.: Cryptographic assumptions: A position paper. In: Kushilevitz, E., Malkin, T. (eds.) Theory of Cryptography. pp. 505–522. Springer Berlin Heidelberg, Berlin, Heidelberg (2016)

[198] Gordon, L.A., Loeb, M.P.: The economics of information security investment. ACM Trans. Inf. Syst. Secur. 5(4), 438–457 (Nov 2002), http://doi.acm.org/10.1145/581271.581274

[199] Goryczka, S., Xiong, L., Sunderam, V.: Secure Multiparty Aggregation with Differential Privacy: A Comparative Study. In: Proceedings of the Joint EDBT/ICDT 2013 Workshops. pp. 155–163. EDBT '13, ACM (2013)

[200] Götzfried, J., Eckert, M., Schinzel, S., Müller, T.: Cache attacks on intel sgx. In: Proceedings of the 10th European Workshop on Systems Security. pp. 2:1–2:6. EuroSec'17, ACM, New York, NY, USA (2017), http://doi.acm.org/10.1145/3065913.3065915

[201] Gramaglia, M., Fiore, M., Tarable, A., Banchs, A.: $k\tau, \epsilon$-anonymity: Towards privacy-preserving publishing of spatiotemporal trajectory data. CoRR abs/1701.02243 (2017)

[202] Greene, J.: Whitepaper: Intel trusted execution technology. https://www.intel.com/content/www/us/en/architecture-and-technology/trusted-execution-technology/trusted-execution-technology-security-paper.html (2012)

[203] Grontas, P.: Secure multi party computations for electronic voting. http://mpla.math.uoa.gr/media/theses/msc/P.%20Grontas.pdf (2014)

[204] Group, T.C.: Trusted computing platform alliance, main specification version 1.1b. https://trustedcomputinggroup.org/wp-content/uploads/TCPA_Main_TCG_Architecture_v1_1b.pdf (2002)

[205] Group, T.C.: Tcg specification, architecture overview. https://trustedcomputinggroup.org/wp-content/uploads/TCG_1_4_Architecture_Overview.pdf (2007)

[206] Group, T.C.: TPM Main Specification, Part 1: Design Principles. https://trustedcomputinggroup.org/wp-content/uploads/TPM-Main-Part-1-Design-Principles_v1.2_rev116_01032011.pdf (2011)

[207] Group, T.C.: TPM Main Specification, Part 2: TPM Structures. https://trustedcomputinggroup.org/wp-content/uploads/TPM-Main-Part-2-TPM-Structures_v1.2_rev116_01032011.pdf (2011)

[208] Group, T.C.: TPM Main Specification, Part 3: Commands. https://trustedcomputinggroup.org/wp-content/uploads/TPM-Main-Part-3-Commands_v1.2_rev116_01032011.pdf (2011)

[209] Group, T.C.: Trusted computing group glossary. https://trustedcomputinggroup.org/wp-content/uploads/TCG_Glossary_Board-Approved_12.13.2012.pdf (2012)

[210] Group, T.C.: Trusted platform module library, Part 1: Architecture. https://trustedcomputinggroup.org/wp-content/uploads/TPM-Rev-2.0-Part-1-Architecture-01.38.pdf (2016)

[211] Gruss, D.: Software-based Microarchitectural Attacks. Ph.D. thesis (2017)

[212] Gruss, D.: Software-based microarchitectural attacks. it - Information Technology (60), 335–341 (2018)

[213] Gruss, D., Maurice, C., Wagner, K., Mangard, S.: Flush+flush: A fast and stealthy cache attack. In: Caballero, J., Zurutuza, U., Rodríguez, R.J. (eds.) Detection of Intrusions and Malware, and Vulnerability Assessment. pp. 279–299. Springer International Publishing, Cham (2016)

[214] Gueron, S.: Memory encryption for general-purpose processors. IEEE Security Privacy 14(6), 54–62 (Nov 2016)

[215] Haeberlen, A., Pierce, B.C., Narayan, A.: Differential privacy under fire. In: Proceedings of the 20th USENIX Security Symposium (Aug 2011)

[216] Hahn, F., Kerschbaum, F., Agnes, K., Schneider, T., Zohner, M., Pullonen, P., Orlandi, C.: The full set of new protocols. Tech. Rep. D13.3, BIU (2016)

[217] Halevi, S., Shoup, V.: Algorithms in helib. In: Garay, J.A., Gennaro, R. (eds.) Advances in Cryptology – CRYPTO 2014. pp. 554–571. Springer Berlin Heidelberg, Berlin, Heidelberg (2014)

[218] Hall, R., Rinaldo, A., Wasserman, L.: Random Differential Privacy. ArXiv e-prints (Dec 2011)

[219] Han, J., Kamber, M., Pei, J.: Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edn. (2011)

[220] Hankerson, D., Menezes, A.: Elliptic Curve Cryptography, pp. 397–397. Springer US, Boston, MA (2011), https://doi.org/10.1007/978-1-4419-5906-5_245

[221] Hart, G.: Nonintrusive Appliance Load Monitoring. Proc. of the IEEE 80(12), 1870–1891 (1992)

[222] Hastings, M., Hemenway, B., Noble, D., Zdancewic, S.: Sok: General purpose compilers for secure multi-party computation. In: 2019 IEEE Symposium on Security and Privacy (SP). pp. 1220–1237 (May 2019)

[223] Hazay, C., Lindell, Y.: A note on the relation between the definitions of security for semi-honest and malicious adversaries. Cryptology ePrint Archive, Report 2010/551 (2010), https://eprint.iacr.org/2010/551

[224] Hazay, C., Venkitasubramaniam, M.: Scalable multi-party private set-intersection. In: Fehr, S. (ed.) Public-Key Cryptography – PKC 2017. pp. 175–203. Springer Berlin Heidelberg, Berlin, Heidelberg (2017)

[225] He, W., Zhang, W., Das, S., Liu, Y.: Sgxlinger: A new side-channel attack vector based on interrupt latency against enclave execution. In: 2018 IEEE 36th International Conference on Computer Design (ICCD). pp. 108–114 (Oct 2018)

[226] Henecka, W., Kögl, S., Sadeghi, A., Schneider, T., Wehrenberg, I.: TASTY: Tool for Automating Secure Two-party Computations. In: Proceedings of the 17th ACM Conference on Computer and Communications Security. pp. 451–462. CCS 2010, ACM (2010)

[227] Hirt, M., Maurer, U., Przydatek, B.: Efficient Secure Multi-party Computation. In: Okamoto, T. (ed.) Proc. of the 6th Int. Conf. on the Theory and Application of Cryptology and Information Security - ASIACRYPT 2000. Lecture Notes in C.S., vol. 1976, pp. 143–161. Springer (2000)

[228] Hirt, M.: Multi-Party Computation:Efficient Protocols, General Adversaries,and Voting. Ph.D. thesis, ETH Zuerich (2001)

[229] Hoekstra, M.: Intel sgx for dummies (intel sgx design objectives). https://software.intel.com/en-us/blogs/2013/09/26/protecting-application-secrets-with-intel-sgx (2013)

[230] Hoekstra, M.: Intel sgx for dummies – part 2. https://software.intel.com/en-us/blogs/2014/06/02/intel-sgx-for-dummies-part-2 (2014)

[231] Hoekstra, M.: Intel sgx for dummies – part 3. https://software.intel.com/en-us/blogs/2014/09/01/intel-sgx-for-dummies-part-3 (2014)

[232] Hoekstra, M., Lal, R., Pappachan, P., Phegade, V., Del Cuvillo, J.: Using innovative instructions to create trustworthy software solutions. In: Proceedings of the 2Nd International Workshop on Hardware and Architectural Support for Security and Privacy. pp. 11:1–11:1. HASP '13, ACM, New York, NY, USA (2013), http://doi.acm.org/10.1145/2487726.2488370

[233] Hoepman, J.H., Huitema, G.: Privacy enhanced fraud resistant road pricing. In: Berleur, J., Hercheui, M.D., Hilty, L.M. (eds.) What Kind of Information Society? Governance, Virtuality, Surveillance, Sustainability, Resilience. pp. 202–213. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)

[234] Hoofnagle, C.J., Urban, J.M.: Alan westin's privacy homo economicus. Wake Forest Law Review 49(2434800) (2014)

[235] Hsu, J., Gaboardi, M., Haeberlen, A., Khanna, S., Narayan, A., Pierce, B.C., Roth, A.: Differential privacy: An economic method for choosing epsilon. In: Proceedings of the 2014 IEEE 27th Computer Security Foundations Symposium. pp. 398–410. CSF '14, IEEE Computer Society, Washington, DC, USA (2014), http://dx.doi.org/10.1109/CSF.2014.35

[236] Huang, Y., Evans, D., Katz, J., Malka, L.: Faster Secure Two-party Computation Using Garbled Circuits. In: Proceedings of the 20th USENIX Conference on Security. pp. 35–35. SEC 2011, USENIX Association (2011)

[237] Huang, Z.: Privacy Preserving Auction, pp. 1–6. Springer Berlin Heidelberg, Berlin, Heidelberg (2014), https://doi.org/10.1007/978-3-642-27848-8_791-1

[238] IBM: 10 key marketing trends for 2017 and ideas for exceeding customer expectations. https://public.dhe.ibm.com/common/ssi/ecm/wr/en/wrl12345usen/watson-customer-engagement-watson-marketing-wr-other-papers-and-reports-wrl12345usen-20170719.pdf (2017)

[239] (IETF), I.E.T.F.: Randomness requirements for security. Tech. Rep. 4086 (2005)

[240] (IETF), I.E.T.F.: The transport layer security (tls) protocol version 1.3. Tech. Rep. 8446 (2018)

[241] Inc., A.: Differential privacy. https://www.apple.com/privacy/docs/Differential_Privacy_Overview.pdf (2017)

[242] Inoue, A., Iwata, T., Minematsu, K., Poettering, B.: Cryptanalysis of ocb2: Attacks on authenticity and confidentiality. Cryptology ePrint Archive, Report 2019/311 (2019), https://eprint.iacr.org/2019/311

[243] Irazoqui, G., Eisenbarth, T., Sunar, B.: Cross processor cache attacks. In: Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security. pp. 353–364. ASIA CCS 16, ACM, New York, NY, USA (2016), http://doi.acm.org/10.1145/2897845.2897867

[244] Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending oblivious transfers efficiently. In: In Proccedings of the CRYPTO 2003 (LNCS 2729). pp. 145–161. CRYPTO 2003, Springer (2003)

[245] Ishai, Y., Kushilevitz, E., Lindell, Y., Petrank, E.: Black-box constructions for secure computation. In: Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing. p. 99–108. STOC 06, Association for Computing Machinery, New York, NY, USA (2006)

[246] Iyengar, V.S.: Transforming data to satisfy privacy constraints. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 279–288. KDD '02, ACM, New York, NY, USA (2002), http://doi.acm.org/10.1145/775047.775089

[247] Jain, P., Desai, S.J., Shih, M.W., Kim, T., Kim, S.M., Lee, J.H., Choi, C., Shin, Y., Kang, B.B., Han, D.: Opensgx: An open platform for sgx research. In: NDSS. vol. 16, pp. 21–24 (2016)

[248] Jang, Y., Lee, J., Lee, S., Kim, T.: Sgx-bomb: Locking down the processor via rowhammer attack. In: Proceedings of the 2Nd Workshop on System Software for Trusted Execution. pp. 5:1–5:6. SysTEX'17, ACM, New York, NY, USA (2017), http://doi.acm.org/10.1145/3152701.3152709

[249] Jannach, D., Zanker, M., Felfernig, A., Friedrich, G.: Recommender Systems: An Introduction. Cambridge University Press, New York, NY, USA, 1st edn. (2010)

[250] Jardí-Cedó, R., Castellà-Roca, J., Viejo, A.: Privacy-preserving electronic toll system with dynamic pricing for low emission zones. In: Garcia-Alfaro, J., Herrera-Joancomartí, J., Lupu, E., Posegga, J., Aldini, A., Martinelli, F., Suri, N. (eds.) Data Privacy Management, Autonomous Spontaneous Security, and Security Assurance. pp. 327–334. Springer International Publishing, Cham (2015)

[251] Jawurek, M., Kerschbaum, F.a.: Privacy technologies for smart grids - a survey of options. Tech. rep. (November 2012)

[252] Jha, S., Kruger, L., Shmatikov, V.: Towards practical privacy for genomic computation. In: Proceedings of the 2008 IEEE Symposium on Security and Privacy. pp. 216–230. SP '08, IEEE Computer Society, Washington, DC, USA (2008), https://doi.org/10.1109/SP.2008.34

[253] Johnson, S., Scarlata, V., Rozas, C., Brickell, E., McKeen, F.: Epid provisioning and attestation services. https://software.intel.com/sites/default/files/managed/57/0e/ww10-2016-sgx-provisioning-and-attestation-final.pdf (2016)

[254] Jorgensen, Z., Yu, T., Cormode, G.: Conservative or liberal? personalized differential privacy. In: 2015 IEEE 31st International Conference on Data Engineering (ICDE). pp. 1023–1034. IEEE Computer Society, Los Alamitos, CA, USA (apr 2015), https://doi.ieeecomputersociety.org/10.1109/ICDE.2015.7113353

[255] Jutla, C.S.: Encryption modes with almost free message integrity. In: Pfitzmann, B. (ed.) Advances in Cryptology — EUROCRYPT 2001. pp. 529–544. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)

[256] Kairouz, P., Oh, S., Viswanath, P.: Secure Multi-party Differential Privacy. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems 28, pp. 2008–2016. Curran Associates (2015)

[257] Kantarcioglu, M.: A Survey of Privacy-Preserving Methods Across Horizontally Partitioned Data. In: Aggarwal, C., Yu, P. (eds.) Privacy-Preserving Data Mining: Models and Algorithms. pp. 313–335. Springer (2008)

[258] Kantarcioglu, M., Kardes, O.: Privacy-Preserving Data Mining in the Malicious Model. Int. J. Inf. Comput. Secur. 2(4), 353–375 (2008)

[259] Kargupta, H., Das, K., Liu, K.: Multi-party, Privacy-Preserving Distributed Data Mining Using a Game Theoretic Framework, pp. 523–531. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)

[260] Katz, J., Lindell, Y.: Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series). Chapman & Hall/CRC (2007)

[261] Keromytis, A.D.: Buffer Overflow Attacks, pp. 174–177. Springer US, Boston, MA (2011), https://doi.org/10.1007/978-1-4419-5906-5_502

[262] Kifer, D.: Attacks on privacy and definetti's theorem. pp. 127–138. SIGMOD '09, ACM (2009)

[263] Kifer, D., Lin, B.R.: Towards an axiomatization of statistical privacy and utility. In: Proceedings of the Twenty-ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. pp. 147–158. PODS '10, ACM, New York, NY, USA (2010), http://doi.acm.org/10.1145/1807085.1807106

[264] Kifer, D., Machanavajjhala, A.: No free lunch in data privacy. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data. pp. 193–204. SIGMOD '11, ACM, New York, NY, USA (2011), http://doi.acm.org/10.1145/1989323.1989345

[265] Kifer, D., Machanavajjhala, A.: Pufferfish: A framework for mathematical privacy definitions. ACM Trans. Database Syst. 39(1), 3:1–3:36 (2014)

[266] Kim, S., Shin, Y., Ha, J., Kim, T., Han, D.: A first step towards leveraging commodity trusted execution environments for network applications. In: Proceedings of the 14th ACM Workshop on Hot Topics in Networks. pp. 1–7 (2015)

[267] Kim, Y., Daly, R., Kim, J., Fallin, C., Lee, J.H., Lee, D., Wilkerson, C., Lai, K., Mutlu, O.: Flipping bits in memory without accessing them: An experimental study of dram disturbance errors. In: Proceeding of the 41st Annual International Symposium on Computer Architecuture. pp. 361–372. ISCA '14, IEEE Press, Piscataway, NJ, USA (2014), http://dl.acm.org/citation.cfm?id=2665671.2665726

[268] Kiriansky, V., Waldspurger, C.: Speculative buffer overflows: Attacks and defenses. CoRR abs/1807.03757 (2018), http://arxiv.org/abs/1807.03757

[269] Kissner, L., Song, D.: Privacy-preserving set operations. In: Proceedings of the 25th Annual International Conference on Advances in Cryptology. pp. 241–257. CRYPTO'05, Springer-Verlag, Berlin, Heidelberg (2005), http://dx.doi.org/10.1007/11535218_15

[270] Kluczniak, K.: Domain-specific pseudonymous signatures revisited. Cryptology ePrint Archive, Report 2016/070 (2016), https://eprint.iacr.org/2016/070

[271] Kocher, P., Horn, J., Fogh, A., , Genkin, D., Gruss, D., Haas, W., Hamburg, M., Lipp, M., Mangard, S., Prescher, T., Schwarz, M., Yarom, Y.: Spectre attacks:

Exploiting speculative execution. In: 40th IEEE Symposium on Security and Privacy (S&P'19) (2019)

[272] Kolesnikov, V., Schneider, T.: Improved garbled circuit: Free xor gates and applications. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) Automata, Languages and Programming. pp. 486–498. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)

[273] Küçük, K.A., Paverd, A., Martin, A., Asokan, N., Simpson, A., Ankele, R.: Exploring the Use of Intel SGX for Secure Many-Party Applications. pp. 5:1–5:6. SysTEX 2016, ACM (2016)

[274] Kumaraguru, P., Cranor, L.F.: Privacy Indexes: A Survey of Westin's Studies. Tech. Rep. CMU-ISRI-5-138, Institute for Software Research International, School of Computer Science, Carnegie Mellon University (Dezember 2005)

[275] Kuvaiskii, D., Oleksenko, O., Arnautov, S., Trach, B., Bhatotia, P., Felber, P., Fetzer, C.: Sgxbounds: Memory safety for shielded execution. In: Proceedings of the Twelfth European Conference on Computer Systems. pp. 205–221. EuroSys '17, ACM, New York, NY, USA (2017), http://doi.acm.org/10.1145/3064176.3064192

[276] Kwong, A., Genkin, D., Gruss, D., Yarom, Y.: Rambleed: Reading bits in memory without accessing them. In: 41st IEEE Symposium on Security and Privacy (S&P) (2020)

[277] Laney, D.: 3-d data management: Controlling data volume, velocity, and variety. META Group Res Note 6 6 (01 2001)

[278] Laud, P., Pankova, A., Kamm, L., Veeningen, M.: Basic constructions of secure multiparty computation, pp. 1–25. Cryptology and Information Security Series, IOS Press, Netherlands (2015)

[279] Launchbury, J., Archer, D., DuBuisson, T., Mertens, E.: Application-scale secure multiparty computation. In: Shao, Z. (ed.) Programming Languages and Systems. pp. 8–26. Springer Berlin Heidelberg, Berlin, Heidelberg (2014)

[280] Lee, J., Clifton, C.: How much is enough? choosing ϵ for differential privacy. In: Proceedings of the 14th International Conference on Information Security. pp. 325–340. ISC'11, Springer-Verlag, Berlin, Heidelberg (2011), http://dl.acm.org/citation.cfm?id=2051002.2051032

[281] Lee, S., Shih, M.W., Gera, P., Kim, T., Kim, H., Peinado, M.: Inferring fine-grained control flow inside SGX enclaves with branch shadowing. In: 26th USENIX Security Symposium (USENIX Security 17). pp. 557–574. USENIX Association, Vancouver, BC (2017), https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/lee-sangho

[282] Li, N., Li, T., Venkatasubramanian, S.: t-closeness: Privacy beyond k-anonymity and l-diversity. In: 2007 IEEE 23rd International Conference on Data Engineering. pp. 106–115 (April 2007)

[283] Li, N., Qardaji, W., Su, D., Wu, Y., Yang, W.: Membership privacy: A unifying framework for privacy definitions. pp. 889–900. CCS '13, ACM (2013)

[284] Lind, J., Priebe, C., Muthukumaran, D., O'Keeffe, D., Aublin, P.L., Kelbert, F., Reiher, T., Goltzsche, D., Eyers, D., Kapitza, R., Fetzer, C., Pietzuch, P.: Glamdring: Automatic application partitioning for intel SGX. In: 2017 USENIX Annual Technical Conference (USENIX ATC 17). pp. 285–298. USENIX Association, Santa Clara, CA (2017), https://www.usenix.org/conference/atc17/technical-sessions/presentation/lind

[285] Lindell, Pinkas: Privacy preserving data mining. Journal of Cryptology 15(3), 177–206 (Jun 2002)

[286] Lindell, Y., Pinkas, B.: Secure Two-Party Computation via Cut-and-Choose Oblivious Transfer. In: Ishai, Y. (ed.) Proc. of the 8th Theory of Cryptography Conf. - TCC 2011. Lecture Notes in C.S., vol. 6597, pp. 329–346. Springer (2011)

[287] Lindell, Y.: Anonymous authentication. Journal of Privacy and Confidentiality 2(2) (2011)

[288] Lindell, Y.: How to simulate it - a tutorial on the simulation proof technique. Cryptology ePrint Archive, Report 2016/046 (2016), http://eprint.iacr.org/2016/046

[289] Lindell, Y., Pinkas, B.: Privacy Preserving Data Mining. In: Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology. pp. 36–54. CRYPTO '00, Springer (2000)

[290] Lindell, Y., Pinkas, B.: Secure multiparty computation for privacy-preserving data mining. Cryptology ePrint Archive, Report 2008/197 (2008), https://eprint.iacr.org/2008/197

[291] Lipp, M., Schwarz, M., Gruss, D., Prescher, T., Haas, W., Fogh, A., Horn, J., Mangard, S., Kocher, P., Genkin, D., Yarom, Y., Hamburg, M.: Meltdown: Reading kernel memory from user space. In: 27th USENIX Security Symposium (USENIX Security 18) (2018)

[292] Liu, C., Chakraborty, S., Mittal, P.: Dependence makes you vulnerable: Differential privacy under dependent tuples. In: Proceedings of the The Network and Distributed System Security Symposium — NDSS 2016 (2016)

[293] Liu, K., Giannella, C., Kargupta, H.: A Survey of Attack Techniques on Privacy-Preserving Data Perturbation Methods, pp. 359–381. Springer US, Boston, MA (2008), https://doi.org/10.1007/978-0-387-70992-5_15

[294] Liu, Y., Simpson, A.C.: Privacy-preserving targeted mobile advertising: Requirements, design, and a prototype implementation. Software: Practice and Experience 46(12), 1657–1684 (2016)

[295] Lo, C., Huang, C., Ku, J.: A cooperative intrusion detection system framework for cloud computing networks. In: 2010 39th International Conference on Parallel Processing Workshops. pp. 280–284 (Sep 2010)

[296] Ma, Q., Deng, P.: Secure multi-party protocols for privacy preserving data mining. In: Li, Y., Huynh, D.T., Das, S.K., Du, D.Z. (eds.) Wireless Algorithms, Systems, and Applications. pp. 526–537. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)

[297] Machanavajjhala, A., Kifer, D., Abowd, J., Gehrke, J., Vilhuber, L.: Privacy: Theory meets practice on the map. In: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering. pp. 277–286. ICDE '08, IEEE Computer Society, Washington, DC, USA (2008), https://doi.org/10.1109/ICDE.2008.4497436

[298] Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M.: L-diversity: Privacy beyond k-anonymity. ACM Trans. Knowl. Discov. Data 1(1) (Mar 2007), http://doi.acm.org/10.1145/1217299.1217302

[299] Machluf, Y., Tal, O., Navon, A., Chaiter, Y.: From population databases to research and informed health decisions and policy. Front Public Health 5(230) (2017)

[300] Malik, M.B., Ghazi, M.A., Ali, R.: Privacy preserving data mining techniques: Current scenario and future prospects. In: 2012 Third International Conference on Computer and Communication Technology. pp. 26–32 (Nov 2012)

[301] Malkhi, D., Nisan, N., Pinkas, B., Sella, Y.: Fairplay — A Secure Two-Party Computation System. In: Proceedings of the 13th Conference on USENIX Security Symposium. pp. 20–20. SSYM 2004, USENIX Association (2004)

[302] Marr, B.: Big data: 20 mind-boggling facts everyone must read. https://www.forbes.com/sites/bernardmarr/2015/09/30/big-data-20-mind-boggling-facts-everyone-must-read/#5fdcb9ac17b1 (2015)

[303] Martin, A.: The Ten Page Introduction to Trusted Computing. Tech. Rep. RR-08-11, University of Oxford (2008)

[304] Martins, P., Sousa, L., Mariano, A.: A survey on fully homomorphic encryption: An engineering perspective. ACM Comput. Surv. 50(6) (Dec 2017), https://doi.org/10.1145/3124441

[305] Marton, K., Suciu, A., Ignat, I.: Randomness in digital cryptography: A survey. Romanian Journal of Information Science and Technology 13 (01 2010)

[306] Matetic, S., Ahmed, M., Kostiainen, K., Dhar, A., Sommer, D., Gervais, A., Juels, A., Capkun, S.: ROTE: Rollback protection for trusted execution. In: 26th USENIX Security Symposium (USENIX Security 17). pp. 1289–1306. USENIX Association, Vancouver, BC (2017), https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/matetic

[307] Mazloom, S., Gordon, S.D.: Secure computation with differentially private access patterns. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. pp. 490–507. CCS '18, ACM, New York, NY, USA (2018), http://doi.acm.org/10.1145/3243734.3243851

[308] Mccormick, T., Lee, H., Cesare, N., Shojaie, A., Spiro, E.: Using twitter for demographic and social science research: Tools for data collection and processing. Sociological Methods and Research (09 2015)

[309] McCune, J.M., Parno, B.J., Perrig, A., Reiter, M.K., Isozaki, H.: Flicker: An execution infrastructure for tcb minimization. SIGOPS Oper. Syst. Rev. 42(4), 315–328 (Apr 2008), http://doi.acm.org/10.1145/1357010.1352625

[310] McKeen, F., Alexandrovich, I., Berenzon, A., Rozas, C.V., Shafi, H., Shanbhogue, V., Savagaonkar, U.R.: Innovative instructions and software model for isolated execution. In: Proceedings of the 2Nd International Workshop on Hardware and Architectural Support for Security and Privacy. pp. 10:1–10:1. HASP '13, ACM, New York, NY, USA (2013), http://doi.acm.org/10.1145/2487726.2488368

[311] McMahan, H.B., Moore, E., Ramage, D., y Arcas, B.A.: Federated learning of deep networks using model averaging. CoRR abs/1602.05629 (2016), http://arxiv.org/abs/1602.05629

[312] McSherry, F.: Privacy integrated queries: An extensible platform for privacy-preserving data analysis. Commun. ACM 53(9), 89–97 (Sep 2010), http://doi.acm.org/10.1145/1810891.1810916

[313] McSherry, F.: Lunchtime for data privacy. https://github.com/frankmcsherry/blog/blob/master/posts/2016-08-16.md (2016)

[314] McSherry, F., Talwar, K.: Mechanism design via differential privacy. In: Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science. pp. 94–103. FOCS '07, IEEE Computer Society, Washington, DC, USA (2007), http://dx.doi.org/10.1109/FOCS.2007.41

[315] Meiklejohn, S., Mowery, K., Checkoway, S., Shacham, H.: The phantom tollbooth: Privacy-preserving electronic toll collection in the presence of driver collusion. In: Proceedings of the 20th USENIX Conference on Security. pp. 32–32. SEC'11, USENIX Association, Berkeley, CA, USA (2011), http://dl.acm.org/citation.cfm?id=2028067.2028099

[316] Mendes, R., Vilela, J.P.: Privacy-preserving data mining: Methods, metrics, and applications. IEEE Access 5, 10562–10582 (2017)

[317] Menezes, A.J., Vanstone, S.A., Oorschot, P.C.V.: Handbook of Applied Cryptography. CRC Press, Inc., Boca Raton, FL, USA, 1st edn. (1996)

[318] Mironov, I.: Renyi Differential Privacy. ArXiv e-prints (Feb 2017)

[319] Mironov, I., Pandey, O., Reingold, O., Vadhan, S.: Computational differential privacy. In: Halevi, S. (ed.) Advances in Cryptology - CRYPTO 2009. pp. 126–142. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)

[320] Miyaji, A., Rahman, M.S.: Privacy-Preserving Data Mining: A Game-Theoretic Approach, pp. 186–200. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)

[321] Moghimi, A., Irazoqui, G., Eisenbarth, T.: Cachezoom: How sgx amplifies the power of cache attacks. In: Fischer, W., Homma, N. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2017. pp. 69–90. Springer International Publishing, Cham (2017)

[322] Mohan, P., Thakurta, A., Shi, E., Song, D., Culler, D.: Gupt: Privacy preserving data analysis made easy. In: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. pp. 349–360. SIGMOD '12, ACM, New York, NY, USA (2012), http://doi.acm.org/10.1145/2213836.2213876

[323] Montenegro, J.A., Fischer, M.J., Lopez, J., Peralta, R.: Secure sealed-bid online auctions using discreet cryptographic proofs. Mathematical and Computer Modelling 57(11), 2583 – 2595 (2013), http://www.sciencedirect.com/science/article/pii/S0895717711004535, information System Security and Performance Modeling and Simulation for Future Mobile Networks

[324] Morais, E., van Wijk, C., Koens, T.: Zero knowledge set membership. https://www.ingwb.com/media/2667856/zero-knowledge-set-membership.pdf (2018)

[325] Nair, D.G., Binu, V.P., Santhosh Kumar, G.: An Improved E-voting scheme using Secret Sharing based Secure Multi-party Computation. arXiv e-prints arXiv:1502.07469 (Feb 2015)

[326] Naor, M.: On cryptographic assumptions and challenges. In: Boneh, D. (ed.) Advances in Cryptology - CRYPTO 2003. pp. 96–109. Springer Berlin Heidelberg, Berlin, Heidelberg (2003)

[327] Naor, M., Pinkas, B., Sumner, R.: Privacy preserving auctions and mechanism design. In: Proceedings of the 1st ACM Conference on Electronic Commerce. pp. 129–139. EC '99, ACM, New York, NY, USA (1999), http://doi.acm.org/10.1145/336992.337028

[328] Narayan, A., Haeberlen, A.: Djoin: Differentially private join queries over distributed databases. In: Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation. pp. 149–162. OSDI'12, USENIX Association, Berkeley, CA, USA (2012), http://dl.acm.org/citation.cfm?id=2387880.2387895

[329] Narayanan, A., Shmatikov, V.: How to break anonymity of the netflix prize dataset. CoRR abs/cs/0610105 (2006), http://arxiv.org/abs/cs/0610105

[330] Nergiz, M.E., Atzori, M., Clifton, C.: Hiding the presence of individuals from shared databases. In: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data. pp. 665–676. SIGMOD '07, ACM, New York, NY, USA (2007), http://doi.acm.org/10.1145/1247480.1247554

[331] Nergiz, M.E., Clifton, C.: δ presence without complete world knowledge. IEEE Trans. on Knowl. and Data Eng. 22(6), 868–883 (Jun 2010), http://dx.doi.org/10.1109/TKDE.2009.125

[332] Nergiz, M.E., Clifton, C., Nergiz, A.E.: Multirelational k-anonymity. IEEE Trans. on Knowl. and Data Eng. 21(8), 1104–1117 (Aug 2009), http://dx.doi.org/10.1109/TKDE.2008.210

[333] NIST: Announcing the advanced encryption standard (aes). Tech. Rep. FIPS.197, NIST (2001)

[334] NIST: Guidlines on Cryptographic Algorithms Usage and Key Management. Tech. Rep. EPC342-08, NIST (2016)

[335] NIST: NIST Cryptographic Standards and Guidelines Development Process. Tech. Rep. NISTIR 7977, NIST (2016)

[336] OECD: The oecd privacy framework. http://www.oecd.org/sti/ieconomy/oecd_privacy_framework.pdf (2013)

[337] O'Hara, K., Whitley, E., Whittall, P.: Avoiding the jigsaw effect: Experiences with ministry of justice reoffending data (December 2011)

[338] Ohm, P.: Broken promises of privacy: Responding to the surprising failure of anonymization. UCLA Law Review 57, 1701–1778 (2010)

[339] Oleksenko, O., Trach, B., Krahn, R., Silberstein, M., Fetzer, C.: Varys: Protecting SGX enclaves from practical side-channel attacks. In: 2018 USENIX Annual Technical Conference (USENIX ATC 18). pp. 227–240. USENIX Association, Boston, MA (2018), https://www.usenix.org/conference/atc18/presentation/oleksenko

[340] Orenbach, M., Lifshits, P., Minkin, M., Silberstein, M.: Eleos: Exitless os services for sgx enclaves. In: Proceedings of the Twelfth European Conference on Computer Systems. pp. 238–253. EuroSys '17, ACM, New York, NY, USA (2017), http://doi.acm.org/10.1145/3064176.3064219

[341] Orlandi, C.: Is multiparty computation any good in practice? pp. 5848–5851 (05 2011)

[342] Orlandi, C., Pinkas, B., Warinschi, B., Bogdanov, D., Scheider, T., Zohner, M., Veeningen, M., de Vreede, N.: A theoretical evaluation of the existing secure computation solutions. Tech. Rep. D11.1, BIU (2015)

[343] Osadchy, M., Pinkas, B., Jarrous, A., Moskovich, B.: Scifi - a system for secure face identification. In: 2010 IEEE Symposium on Security and Privacy. pp. 239–254 (May 2010)

[344] Pai, M.M., Roth, A.: Privacy and mechanism design. SIGecom Exch. 12(1), 8–29 (Jun 2013), http://doi.acm.org/10.1145/2509013.2509016

[345] Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) Advances in Cryptology — EUROCRYPT 99. pp. 223–238. Springer Berlin Heidelberg, Berlin, Heidelberg (1999)

[346] Parno, B., Lorch, J., Douceur, J.J., Mickens, J., McCune, J.M.: Memoir: Practical state continuity for protected modules. In: Proceedings of the IEEE Symposium

on Security and Privacy. IEEE (May 2011), https://www.microsoft.com/en-us/research/publication/memoir-practical-state-continuity-for-protected-modules/

[347] Patel, A., Qassim, Q., Wills, C.: A survey of intrusion detection and prevention systems. Information Management & Computer Security 18(4), 277–290 (2010)

[348] Pattuk, E., Kantarcioglu, M., Ulusoy, H., Malin, B.: Cheapsmc: A framework to minimize secure multiparty computation cost in the cloud. In: Ranise, S., Swarup, V. (eds.) Data and Applications Security and Privacy XXX. pp. 285–294. Springer International Publishing, Cham (2016)

[349] Paverd, A., Martin, A., Brown, I.: Modelling and automatically analysing privacy properties for honest-but-curious adversaries. https://www.cs.ox.ac.uk/people/andrew.paverd/casper/casper-privacy-report.pdf (2014)

[350] Paverd, A.: Trustworthy remote entities in the smart grid. In: 28th ACM Symposium On Applied Computing - SAC2013. Coimbra, Portugal (2013), http://dl.acm.org/citation.cfm?id=2480362, student Research Competition Finalist

[351] Paverd, A.: Enhancing Communication Privacy Using Trustworthy Remote Entities (2016)

[352] Paverd, A.J., Martin, A.P., Brown, I.: Privacy-enhanced bi-directional communication in the smart grid using trusted computing. In: Fifth IEEE International Conference on Smart Grid Communications (SmartGridComm 2014) (2014), http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=7007758

[353] Paverd, A.J., Martin, A.P., Brown, I.: Security and privacy in smart grid demand response systems. In: Cuellar, J. (ed.) Smart Grid Security, pp. 1–15. Lecture Notes in Computer Science, Springer International Publishing (2014), http://link.springer.com/chapter/10.1007/978-3-319-10329-7_1

[354] Peng, T., Leckie, C., Ramamohanarao, K.: Information sharing for distributed intrusion detection systems. Journal of Network and Computer Applications 30(3), 877 – 899 (2007), http://www.sciencedirect.com/science/article/pii/S1084804505000494

[355] Peng, Z.: Danger of using fully homomorphic encryption: A look at microsoft seal. arXiv preprint arXiv:1906.07127 (2019)

[356] Perla, E., Oldani, M.: Chapter 2 - a taxonomy of kernel vulnerabilities. In: Perla, E., Oldani, M. (eds.) A Guide to Kernel Exploitation, pp. 21 – 45. Syngress, Boston (2011), http://www.sciencedirect.com/science/article/pii/B9781597494861000024

[357] Pettai, M., Laud, P.: Combining differential privacy and secure multiparty computation. In: Proceedings of the 31st Annual Computer Security Applications Conference. pp. 421–430. ACSAC 2015, ACM, New York, NY, USA (2015), http://doi.acm.org/10.1145/2818000.2818027

[358] Pfitzmann, A., Köhntopp, M.: Anonymity, Unobservability, and Pseudonymity — A Proposal for Terminology, pp. 1–9. Springer Berlin Heidelberg, Berlin, Heidelberg (2001), https://doi.org/10.1007/3-540-44702-4_1

[359] Pillitteri, V., Brewer, T.: Guidelines for smart grid cybersecurity. Tech. Rep. 7628 Rev 1, NIST (2014)

[360] Pinkas, B., Schneider, T., Smart, N., Williams, S.: Secure Two-Party Computation Is Practical. ASIACRYPT 2009, vol. 5912, pp. 250–267. Springer (2009)

[361] Pinkas, B.: Cryptographic techniques for privacy-preserving data mining. SIGKDD Explor. Newsl. 4(2), 12–19 (Dec 2002), http://doi.acm.org/10.1145/772862.772865

[362] Post, R.: Three concepts of privacy. Georgetown Law Journal 89(6), 2087–2098 (June 2001)

[363] Proudler, G.: Concepts of Trusted Computing, Applications of Computing, vol. 6, pp. 11–28 (2005)

[364] Quinn, E.L.: Privacy and the new energy infrastructure. Tech. rep. (2009)

[365] Rabin, M.: How to Exchange Secrets by Oblivious Transfer. Tech. Rep. 81, Aiken Computation Laboratory, Harvard University (1981)

[366] Ragab, H., Milburn, A., Razavi, K., Bos, H., Giuffrida, C.: CrossTalk: Speculative Data Leaks Across Cores Are Real. In: Proceedings of the IEEE Symposium on Security & Privacy 2021 (May 2021), intel Bounty Reward

[367] Rahimi, F., Ipakchi, A.: Demand response as a market resource under the smart grid paradigm. IEEE Transactions on Smart Grid 1(1), 82–88 (June 2010)

[368] Ramakrishnan, N., Keller, B.J., Mirza, B.J., Grama, A.Y., Karypis, G.: Privacy risks in recommender systems. IEEE Internet Computing 5(6), 54–62 (Nov 2001), http://dx.doi.org/10.1109/4236.968832

[369] Rassouli, B., Gündüz, D.: On perfect privacy and maximal correlation. CoRR abs/1712.08500 (2017), http://arxiv.org/abs/1712.08500

[370] Rastogi, V., Hay, M., Miklau, G., Suciu, D.: Relationship privacy: Output perturbation for queries with joins. In: Proceedings of the Twenty-eighth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. pp. 107–116. PODS '09, ACM, New York, NY, USA (2009), http://doi.acm.org/10.1145/1559795.1559812

[371] Rastogi, V., Suciu, D., Hong, S.: The boundary between privacy and utility in data publishing. In: Proceedings of the 33rd International Conference on Very Large Data Bases. pp. 531–542. VLDB '07, VLDB Endowment (2007), http://dl.acm.org/citation.cfm?id=1325851.1325913

[372] Redei, G.: Smith–Waterman Algorithm, pp. 1833–1833. Springer Netherlands, Dordrecht (2008), https://doi.org/10.1007/978-1-4020-6754-9_15795

[373] Ringers, S., Verheul, E., Hoepman, J.H.: An efficient self-blindable attribute-based credential scheme. Cryptology ePrint Archive, Report 2017/115 (2017), https://eprint.iacr.org/2017/115

[374] Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM 21(2), 120–126 (Feb 1978), http://doi.acm.org/10.1145/359340.359342

[375] Rogaway, P., Bellare, M., Black, J.: Ocb: A block-cipher mode of operation for efficient authenticated encryption. ACM Trans. Inf. Syst. Secur. 6(3), 365–403 (Aug 2003), http://doi.acm.org/10.1145/937527.937529

[376] Rosen, J.: The Unwanted Gaze: The Destruction of Privacy in America. Random House Inc., New York, NY, USA (2000)

[377] Rossler, T., Leitold, H., Posch, R.: E-voting: A scalable approach using xml and hardware security modules. In: Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'05) on e-Technology, e-Commerce and e-Service. pp. 480–485. EEE '05, IEEE Computer Society, Washington, DC, USA (2005), https://doi.org/10.1109/EEE.2005.63

[378] Roth, A., Roughgarden, T.: Interactive privacy via the median mechanism. In: Proceedings of the Forty-second ACM Symposium on Theory of Computing. pp. 765–774. STOC '10, ACM, New York, NY, USA (2010), http://doi.acm.org/10.1145/1806689.1806794

[379] Roy, I., Setty, S., Kilzer, A., Shmatikov, V., Witchel, E.: Airavat: Security and privacy for mapreduce. In: Symposium on Networked Systems Design and Implementation (NSDI). USENIX - Advanced Computing Systems Association (April 2010), https://www.microsoft.com/en-us/research/publication/airavat-security-and-privacy-for-mapreduce/

[380] Ryan, P.Y.A., Bismark, D., Heather, J., Schneider, S., Xia, Z.: Prêt À voter:a voter-verifiable voting system. IEEE Transactions on Information Forensics and Security 4(4), 662–673 (Dec 2009)

[381] Saia, J., Zamani, M.: Recent Results in Scalable Multi-Party Computation. In: Italiano, G., Margaria-Steffen, T., Pokorný, J., Quisquater, J., Wattenhofer, R. (eds.) Proc. of the 41st Int. Conf. on Current Trends in Theory and Practice of C.S. - SOFSEM 2015. Lecture Notes in C.S., vol. 8939, pp. 24–44. Springer (2015)

[382] Saltzer, J.H., Schroeder, M.D.: The protection of information in computer systems. Proceedings of the IEEE 63(9), 1278–1308 (Sep 1975)

[383] Samarati, P.: Protecting respondents' identities in microdata release. IEEE Trans. on Knowl. and Data Eng. 13(6), 1010–1027 (Nov 2001), https://doi.org/10.1109/69.971193

[384] Samarati, P., Sweeney, L.: Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression (1998), http://www.csl.sri.com/papers/sritr-98-04/

[385] Sánchez, D., Domingo-Ferrer, J., Martnez, S.: Co-utile disclosure of private data in social networks. Inf. Sci. 441(C), 50–65 (May 2018)

[386] van Schaik, S., Kwong, A., Genkin, D., Yarom, Y.: SGAxe: How SGX fails in practice. https://sgaxeattack.com/ (2020)

[387] van Schaik, S., Minkin, M., Kwong, A., Genkin, D., Yarom, Y.: CacheOut: Leaking data on Intel CPUs via cache evictions. https://cacheoutattack.com/ (2020)

[388] Schunter, M.: Intel software guard extensions: Introduction and open research challenges. In: Proceedings of the 2016 ACM Workshop on Software PROtection. pp. 1–1. SPRO '16, ACM, New York, NY, USA (2016), http://doi.acm.org/10.1145/2995306.2995307

[389] Schuster, F., Costa, M., Fournet, C., Gkantsidis, C., Peinado, M., Mainar-Ruiz, G., Russinovich, M.: Vc3: Trustworthy data analytics in the cloud using sgx. In: Proceedings of the 2015 IEEE Symposium on Security and Privacy. pp. 38–54. SP '15, IEEE Computer Society, Washington, DC, USA (2015), https://doi.org/10.1109/SP.2015.10

[390] Schwarz, M., Lipp, M., Moghimi, D., Van Bulck, J., Stecklina, J., Prescher, T., Gruss, D.: ZombieLoad: Cross-privilege-boundary data sampling. arXiv:1905.05726 (2019)

[391] Schwarz, M., Weiser, S., Gruss, D.: Practical enclave malware with intel sgx. arXiv.org e-Print archive (2 2019)

[392] Schwarz, M., Weiser, S., Gruss, D., Maurice, C., Mangard, S.: Malware guard extension: Using sgx to conceal cache attacks. In: Polychronakis, M., Meier, M. (eds.) Detection of Intrusions and Malware, and Vulnerability Assessment. pp. 3–24. Springer International Publishing, Cham (2017)

[393] scut, t.t.: Exploiting format string vulnerabilities. https://crypto.stanford.edu/cs155old/cs155-spring08/papers/formatstring-1.2.pdf (2001)

[394] Serjantov, A., Danezis, G.: Towards an Information Theoretic Metric for Anonymity. In: Dingledine, R., Syverson, P. (eds.) Revised Papers of the 2nd International Workshop, PET 2002. pp. 41–53. Springer (2003)

[395] Shamir, A.: How to Share a Secret. Communications of the ACM 22(11), 612–613 (1979)

[396] Shamir, A., Rivest, R.L., Adleman, L.M.: Mental Poker, pp. 37–43. Springer US, Boston, MA (1981), https://doi.org/10.1007/978-1-4684-6686-7_5

[397] Shinde, S., Chua, Z.L., Narayanan, V., Saxena, P.: Preventing Page Faults from Telling Your Secrets. In: Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security. pp. 317–328. ASIA CCS '16, ACM (2016)

[398] Skowron, A., Rauszer, C.: The Discernibility Matrices and Functions in Information Systems, pp. 331–362. Springer Netherlands, Dordrecht (1992), https://doi.org/10.1007/978-94-015-7975-9_21

[399] Smaha, S.E.: Haystack: an intrusion detection system. In: [Proceedings 1988] Fourth Aerospace Computer Security Applications. pp. 37–44 (Sep 1988)

[400] Smart, N., Rijmen, V., Gierlichs, B., Paterson, K., Stam, M., Warinschi, B., Watson, G.: Algorithms, key size and parameters report. Tech. Rep. TP-05-14-084-EN-N, ENISA (2014)

[401] Smart, N., Rijmen, V., Stam, M., Warinschi, B., Watson, G.: Study on cryptographic protocols. Tech. Rep. TP-06-14-085-EN-N, ENISA (2014)

[402] Smyth, B., Ryan, M.D., Chen, L.: Formal analysis of privacy in direct anonymous attestation schemes. Science of Computer Programming 111, 300 – 317 (2015), http://www.sciencedirect.com/science/article/pii/S0167642315000702, special Issue on Automated Verification of Critical Systems (AVoCS 2013)

[403] Solove, D.: Conceptualizing privacy. California Law Review 90, 1087–1157 (2002)

[404] Soria-Comas, J., Domingo-Ferrer, J., Sánchez, D., Megías, D.: Individual differential privacy: A utility-preserving formulation of differential privacy guarantees. IEEE Transactions on Information Forensics and Security 12(6), 1418–1429 (June 2017)

[405] Stefanov, E., van Dijk, M., Shi, E., Fletcher, C., Ren, L., Yu, X., Devadas, S.: Path oram: An extremely simple oblivious ram protocol. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security. pp. 299–310. CCS '13, ACM, New York, NY, USA (2013), http://doi.acm.org/10.1145/2508859.2516660

[406] Strackx, R., Piessens, F.: Ariadne: A minimal approach to state continuity. In: 25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016. pp. 875–892 (2016), https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/strackx

[407] Sweeney, L.: Datafly: a system for providing anonymity in medical data, pp. 356–381. Springer US, Boston, MA (1998), https://doi.org/10.1007/978-0-387-35285-5_22

[408] Sweeney, L.: K-anonymity: A model for protecting privacy. Int. J. Uncertain. Fuzziness Knowl.-Based Syst. 10(5), 557–570 (Oct 2002), http://dx.doi.org/10.1142/S0218488502001648

[409] Tassa, T., Mazza, A., Gionis, A.: k-concealment: An alternative model of k-type anonymity. Trans. Data Privacy 5(1), 189–222 (Apr 2012), http://dl.acm.org/citation.cfm?id=2207141.2207142

[410] Terrovitis, M., Mamoulis, N., Kalnis, P.: Privacy-preserving anonymization of set-valued data. Proc. VLDB Endow. 1(1), 115–125 (Aug 2008), http://dx.doi.org/10.14778/1453856.1453874

[411] Thiebeauld, H., Gagnerot, G., Wurcker, A., Clavier, C.: Scatter : A new dimension in side-channel. Cryptology ePrint Archive, Report 2017/706 (2017), https://eprint.iacr.org/2017/706

[412] Toldsepp, K., Pruulmann-Vengerfeldt, P., Laud, P.: Requirements specification based on the interviews. Tech. Rep. D1.2, UaESMC (2012)

[413] Tramèr, F., Zhang, F., Lin, H., Hubaux, J., Juels, A., Shi, E.: Sealed-glass proofs: Using transparent enclaves to prove and sell knowledge. In: 2017 IEEE European Symposium on Security and Privacy (EuroS P). pp. 19–34 (April 2017)

[414] Treiman, D., Lu, Y., Qi, Y.: New approaches to demographic data collection. Chinese sociological review 44, 56 (04 2012)

[415] Tromba, I.: Secure computation on genetic data. https://pdfs.semanticscholar.org/1583/ffe8fc01275566db95be32944507c3d9e065.pdf (2014)

[416] Troncoso, C., Danezis, G., Kosta, E., Balasch, J., Preneel, B.: Pripayd: Privacy-friendly pay-as-you-drive insurance. IEEE Transactions on Dependable and Secure Computing 8(5), 742–755 (Sep 2011)

[417] Truta, T.M., Campan, A., Meyer, P.: Generating microdata with p-sensitive k-anonymity property. In: Jonker, W., Petković, M. (eds.) Secure Data Management. pp. 124–141. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)

[418] Tsai, C.C., Porter, D.E., Vij, M.: Graphene-sgx: A practical library os for unmodified applications on sgx. In: Proceedings of the 2017 USENIX Conference on Usenix Annual Technical Conference. pp. 645–658. USENIX ATC '17, USENIX Association, Berkeley, CA, USA (2017), http://dl.acm.org/citation.cfm?id=3154690.3154752

[419] Tselentis, D., Yannis, G., Vlahogianni, E.: Innovative insurance schemes: Pay as/how you drive. Transportation Research Procedia 14, 362–371 (12 2016)

[420] UN: International covenant on civil and political rights. http://www.ohchr.org/Documents/ProfessionalInterest/ccpr.pdf (1966)

[421] Vaidya, J., Clifton, C.: Privacy-preserving data mining: why, how, and when. IEEE Security Privacy 2(6), 19–27 (Nov 2004)

[422] Vaidya, J.: A Survey of Privacy-Preserving Methods Across Vertically Partitioned Data. In: Aggarwal, C., Yu, P. (eds.) Privacy-Preserving Data Mining: Models and Algorithms. pp. 337–358. Springer (2008)

[423] Vaidya, J., Zhu, Y.M., Clifton, C.W.: Privacy Preserving Data Mining (Advances in Information Security). Springer-Verlag, Berlin, Heidelberg (2005)

[424] Vakali, A., Angelis, L., Pournara, D.: Internet based auctions: a survey on models and applications. SIGecom Exchanges 2, 6–15 (01 2001)

[425] Van Bulck, J., Piessens, F., Strackx, R.: Nemesis: Studying microarchitectural timing leaks in rudimentary cpu interrupt logic. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. pp. 178–195. CCS '18, ACM, New York, NY, USA (2018), http://doi.acm.org/10.1145/3243734.3243822

[426] Völp, M., Lackorzynski, A., Decouchant, J., Rahli, V., Rocha, F., Esteves-Verissimo, P.: Avoiding leakage and synchronization attacks through enclave-side preemption control. In: Proceedings of the 1st Workshop on System Software for Trusted Execution. pp. 6:1–6:6. SysTEX '16, ACM, New York, NY, USA (2016), http://doi.acm.org/10.1145/3007788.3007794

[427] Wagner, I., Eckhoff, D.: Technical privacy metrics: A systematic survey. ACM Comput. Surv. 51(3), 57:1–57:38 (Jun 2018), http://doi.acm.org/10.1145/3168389

[428] Wang, B., Song, W., Lou, W., Hou, Y.T.: Privacy-preserving pattern matching over encrypted genetic data in cloud computing. In: IEEE INFOCOM 2017 - IEEE Conference on Computer Communications. pp. 1–9 (May 2017)

[429] Wang, J., Cheng, Y., Li, Q., Jiang, Y.: Interface-based side channel attack against intel SGX. CoRR abs/1811.05378 (2018), http://arxiv.org/abs/1811.05378

[430] Wang, K., Xu, Y., Fu, A.W.C., Wong, R.C.W.: Ff-anonymity: When quasi-identifiers are missing. In: 2009 IEEE 25th International Conference on Data Engineering. pp. 1136–1139 (March 2009)

[431] Wang, K., Fung, B.C.M.: Anonymizing sequential releases. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 414–423. KDD '06, ACM, New York, NY, USA (2006), http://doi.acm.org/10.1145/1150402.1150449

[432] Wang, X., Ranellucci, S., Katz, J.: Global-scale secure multiparty computation. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 39–56. CCS '17, ACM, New York, NY, USA (2017), http://doi.acm.org/10.1145/3133956.3133979

[433] Warner, S.L.: Randomized response: A survey technique for eliminating evasive answer bias. Journal of the American Statistical Association 60(309), 63–69 (1965), http://www.jstor.org/stable/2283137

[434] Warren, S.D., Brandeis, L.D.: The right to privacy. Harward Law Review 4(5), 193–220 (December 1890)

[435] Weichbrodt, N., Kurmus, A., Pietzuch, P., Kapitza, R.: Asyncshock: Exploiting synchronisation bugs in intel sgx enclaves. In: Askoxylakis, I., Ioannidis, S., Katsikas, S., Meadows, C. (eds.) Computer Security – ESORICS 2016. pp. 440–457. Springer International Publishing, Cham (2016)

[436] Weiser, S., Werner, M.: Sgxio: Generic trusted i/o path for intel sgx. In: Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy. pp. 261–268. CODASPY '17, ACM, New York, NY, USA (2017), http://doi.acm.org/10.1145/3029806.3029822

[437] Weisse, O., Van Bulck, J., Minkin, M., Genkin, D., Kasikci, B., Piessens, F., Silberstein, M., Strackx, R., Wenisch, T.F., Yarom, Y.: Foreshadow-NG: Breaking the virtual memory abstraction with transient out-of-order execution. Technical report (2018)

[438] Wilson, C., Hargreaves, T., Hauxwell-Baldwin, R.: Benefits and risks of smart home technologies. Energy Policy 103, 72 – 83 (2017), http://www.sciencedirect.com/science/article/pii/S030142151630711X

[439] Wong, R.C.W., Fu, A.W.C., Wang, K., Pei, J.: Minimality attack in privacy preserving data publishing. pp. 543–554. VLDB '07, VLDB (2007)

[440] Wong, R.C.W., Li, J., Fu, A.W.C., Wang, K.: (α, k)-anonymity: An enhanced k-anonymity model for privacy preserving data publishing. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 754–759. KDD '06, ACM, New York, NY, USA (2006), http://doi.acm.org/10.1145/1150402.1150499

[441] Woodruff, A., Pihur, V., Consolvo, S., Brandimarte, L., Acquisti, A.: Would a privacy fundamentalist sell their DNA for $1000...if nothing bad happened as a result? the westin categories, behavioral intentions, and consequences. In: 10th Symposium On Usable Privacy and Security (SOUPS 2014). pp. 1–18. USENIX Association, Menlo Park, CA (Jul 2014)

[442] Xiao, X., Tao, Y.: Personalized privacy preservation. In: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data. pp. 229–240. SIGMOD '06, ACM, New York, NY, USA (2006), http://doi.acm.org/10.1145/1142473.1142500

[443] Xiao, X., Tao, Y.: M-invariance: Towards privacy preserving re-publication of dynamic datasets. In: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data. pp. 689–700. SIGMOD '07, ACM, New York, NY, USA (2007), http://doi.acm.org/10.1145/1247480.1247556

[444] Xing, B.C., Shanahan, M., Leslie-Hurd, R.: Intel® software guard extensions (intel® sgx) software support for dynamic memory allocation inside an enclave. In: Proceedings of the Hardware and Architectural Support for Security and Privacy 2016. pp. 11:1–11:9. HASP 2016, ACM, New York, NY, USA (2016), http://doi.acm.org/10.1145/2948618.2954330

[445] Xu, Y., Cui, W., Peinado, M.: Controlled-channel attacks: Deterministic side channels for untrusted operating systems. In: 2015 IEEE Symposium on Security and Privacy. pp. 640–656 (May 2015)

[446] Yang, Z., Wright, R., Subramaniam, H.: Experimental analysis of a privacy-preservingscalar product protocol. International Journal of Computer Systems Science and Engineering 21(1), 47–52 (2006)

[447] Yao, A.: Protocols for Secure Computations. In: Proceedings of the 23rd Annual Symposium on Foundations of Computer Science. pp. 160–164. SFCS 1982, IEEE Computer Society (1982)

[448] Yarom, Y., Falkner, K.: Flush+reload: A high resolution, low noise, l3 cache side-channel attack. In: Proceedings of the 23rd USENIX Conference on Security Symposium. pp. 719–732. SEC'14, USENIX Association, Berkeley, CA, USA (2014), http://dl.acm.org/citation.cfm?id=2671225.2671271

[449] Younis, Y.A., Kifayat, K., Shi, Q., Askwith, B.: A new prime and probe cache side-channel attack for cloud computing. In: 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing. pp. 1718–1724 (Oct 2015)

[450] Yu, S.: Big privacy: Challenges and opportunities of privacy study in the age of big data. IEEE Access 4, 2751–2763 (2016)

[451] Zhang, B., Wang, N., Jin, H.: Privacy concerns in online recommender systems: Influences of control and user data input. In: Proceedings of the Tenth USENIX Conference on Usable Privacy and Security. pp. 159–173. SOUPS'14, USENIX Association, Berkeley, CA, USA (2014)

[452] Zhang, Q., Koudas, N., Srivastava, D., Yu, T.: Aggregate query answering on anonymized tables. In: 2007 IEEE 23rd International Conference on Data Engineering. pp. 116–125 (April 2007)

[453] Zheng, S., Chetty, M., Feamster, N.: User Perceptions of Privacy in Smart Homes. ArXiv e-prints (Feb 2018)

A EXAMPLES OF PRIVACY-PRESERVING OPERATIONS

In this chapter, we give a continuous example to show the 'effects' of different privacy-preserving operations on plain sensitive data. We consider the following example and apply the operations outlined in Section 2.9.1:

Consider a hospital which is required to release statistical information about their patients. This includes a non-anonymised plain data table displayed in Table 23.

In this respect, Table 23 represents a patient table, where the attribute *disease* is considered to be a sensitive value. The attributes *sex, age, job* are non-sensitive and, in combination, form a quasi-identifier. Moreover, we assume that an observer has access to an external table displayed in Table 24. In Table 24 the attribute *name* is considered to be a sensitive value. The quasi-identifier for this table contain the same attributes as for Table 23. We assume further, that the observer knows/infers from its observations that every person with a record in Table 24 has also an entry in Table 23.

Table 23: Non-anonymised plain data table for our continuous example — patients records.

Disease	Sex	Age	Job
Depression	Male	35	Programmer
Depression	Male	38	Programmer
Stress	Male	38	Security Analyst
Flu	Male	36	Security Analyst
Stress	Female	30	Manager
Stress	Female	30	CEO
Stress	Female	30	Manager

Observation: It is easy to see, that if these two tables are combined over the quasi-identifiers, that, for example, Victor, a 36 year old security analyst, has flu.

A.1 Preserving Privacy via Generalisation

In Table 25 we applied a generalisation using the taxonomy given in Figure 43, resulting in a 3-anonymous table. As the reader can see, each entry of Table (b) can be mapped to at least 3 records in Table 25.

Table 24: Non-anonymised plain data table for our continuous example — external patients records.

Name	Sex	Age	Job
Alice	Female	30	Manager
Bob	Male	35	Programmer
Eve	Female	30	Manager
David	Male	38	Security Analyst
Carol	Female	30	CEO
Oscar	Male	38	Programmer
Mallory	Female	30	CEO
Victor	Male	36	Security Analyst
Peggy	Female	30	Manager

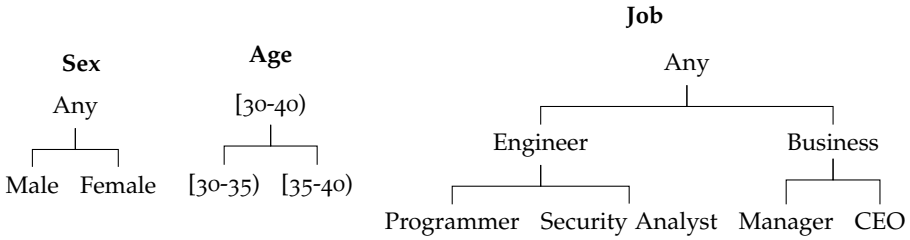


Figure 43: Taxonomy trees for the categories *sex*, *age* and *job* for our continuous example.

Table 25: 3-anonymous table via generalisation.

Disease	Sex	Age	Job
Depression	Male	[35-40)	Engineer
Depression	Male	[35-40)	Engineer
Stress	Male	[35-40)	Engineer
Flu	Male	[35-40)	Engineer
Stress	Female	[30-35)	Business
Stress	Female	[30-35)	Business
Stress	Female	[30-35)	Business

A.2 Preserving Privacy via Suppression

In Table 26 we applied a suppression, resulting in a 2-anonymous table. In row 4, we applied entry suppression on a single entry, while in row 6, we suppressed the whole record further known as record suppression.

A.3 Preserving Privacy via Anatomisation

In Table 27 we applied an anatomisation, resulting in an anonymous table. In this technique, we start from a generalised table (*e.g.* Table 25) and replace the sensitive attribute with a groupID. The groupID is then represented in a separate table, contain-

Table 26: 2-anonymous table via suppression.

Disease	Sex	Age	Job
Depression	Male	[35-40)	Engineer
Depression	Male	[35-40)	Engineer
Stress	Male	[35-40)	Engineer
* * *	Male	[35-40)	Engineer
Stress	Female	[30-35)	Business
* * *	* * *	* * *	* * *
Stress	Female	[30-35)	Business

ing the sensitive value and a counter value mapping the frequency of occurrence in the original table.

Table 27: Anonymous table via anatomisation.

GroupID	Sex	Age	Job
1	Male	35	Programmer
1	Male	38	Programmer
1	Male	38	Security Analyst
1	Male	36	Security Analyst
2	Female	30	Manager
2	Female	30	CEO
2	Female	30	Manager

GroupID	Disease	Count
1	Depression	2
1	Stress	1
1	Flu	1
2	Stress	3

A.4 Preserving Privacy via Permutation

In Table 28 we applied the permutation technique, resulting an anonymous table. This technique operates similar to anatomisation, by grouping the records based on the quasi-identifier and then permuting the entries randomly within the same group. As seen in Table 28, if there is not enough diversity within the same group, this can lead to problems in the anonymisation process.

Table 28: Anonymous table via permutation.

GroupID	Sex	Age	Job
1	Male	35	Programmer
1	Male	38	Programmer
1	Male	38	Security Analyst
1	Male	36	Security Analyst
2	Female	30	Manager
2	Female	30	CEO
2	Female	30	Manager

GroupID	Disease
1	Flu
1	Depression
1	Stress
1	Depression
2	Stress
2	Stress
2	Stress

B TERMS AND DEFINITIONS

In the following, we introduce nomenclature and definitions associated with information privacy which are used in the main matter of this dissertation. In particular, the terminology is focused on the application of privacy-preserving data analysis and data release. As a disclaimer, we note that the terminology and definitions introduced here are informal[1]. Overall the aim is to clarify understanding of these terms for the reader to follow the contributions of this dissertation.

Stakeholders: In any privacy-preserving processing there are multiple stakeholders. Some entities may hold multiple roles at the same time or the roles may be completely distinct. The list presented below is not exhaustive: there may be other entities or stakeholders which we don't list here, but are valid participants in a data processing / release process.

(a) A *data owner*, *record owner*, *data subject* or *PII principal* represents an individual whose data is processed. The data provided by such an individual may be sensitive or non-sensitive. This data may also be associated to an unique identifier, which can link the identity of such an individual to the data [8, 177].

(b) A *data controller* or *PII controller* is an entity who decides on the processing function that is applied on the collected data. As such, this entity is responsible to ensure that the selected processing function is, for example, compliant with good practices for data privacy and, as such, does not leak information about a data subject's sensitive data. The data controller may also act as a data processor [8, 177].

(c) A *trusted analyst*, *data processor* or *PII processor* is an entity who 'executes' the processing function that a data controller has selected. In case these two are separate entities, the data processors acts on behalf of the data controller. A data processor is, in general, also responsible for ensuring privacy-preserving processing [8, 177].

[1] To clarify, we stated informal definitions (throughout the main matter of the dissertation) when either there exists no clear and commonly accepted definition throughout recent literature or such definitions vary widely between different authors. In this regard, the informal definitions hold within the context of this dissertation. Formal definitions are stated where appropriate. (Where found, we support a definition with a reference.)

In the literature, the previously mentioned entities appear under different names and different trust assumptions. The 'behaviour' or underlying structure of the entities remains the same. For example, in the privacy-preserving data analysis and data release literature, common entities are: (a) data providers; (b) computation providers; and (c) data analysts. Data analysts are typically external of the privacy-preserving system and considered to be (potentially) malicious. (This is discussed in more detail in Chapters 3 and 4). In the secure multi-party computation literature common entities are: (a) input parties; (b) computation parties; and (c) output parties. In this model, typically none of the parties trust any other party with their privacy inputs. (This is discussed in more detail in Chapter 5.)

Representation of Data: In any computational task considered in the main matter of the dissertation we are operating on data. Data is represented in a certain structure, for example, as outlined in the following:

(a) An *attribute* [9] is a component of a dataset. Each attribute has a data type (*e.g.* integer, string, double). Attributes may be classified as *sensitive* or *non-sensitive*[2]. In order to preserve an individual's privacy, sensitive attributes are typically removed, altered or in any other way obfuscated.

(b) A *quasi-identifier* [9] is a collection of non-sensitive attributes that can be used to uniquely identify an individual. Thus, to preserve an individual's privacy, it is necessary to identify and potentially obfuscate these quasi-identifiers.

(c) A *domain* is defined as the set of all (unique) values that are permitted for an attribute.

(d) A *tuple* is an entry over all domains of a table. It is typically linked to a certain individual.

(e) A *table* is a well defined structure to abstract a dataset. It consists of at least one or multiple domains as well as at least one or multiple tuples. Tables may be split vertically (each table contains values of a different domain) or horizontally (each table has the same domain, but different tuples).

(f) A *dataset* [9] is a loose collection of data values from a universe. A dataset may be distributed among several entities or reside at a single entity. In a computational context, a dataset is abstracted as one or multiple tables.

(g) An *anonymous dataset* [9] is a dataset where the sensitive attributes are obfuscated.

(h) *Auxiliary information* [9] is the collection of entries of any (public) datasets apart from the 'target'[3] dataset, which may be linked to a entry in the target dataset and, as such, may be used to breach the privacy of this entry.

[2] A sensitive value is typically associated to a property an individual is not willing to publicly disclose (*e.g.* sex, address, earnings). A non-sensitive value are all other values. For a more detailed discussion in this regard we refer the reader to Part I of Chapter 2.

[3] The 'target' dataset is the current dataset which a user interacts with. (By utilising privacy mechanisms we aim to protect the 'target' dataset and, as such, prevent linking attacks using auxiliary information.)

Processing of Data: Data are raw, unorganised and unstructured facts which, when 'processed', can be used to derive information from it. In the following, we define what we understand by 'processing' data:

(a) Under *processing* of information we understand the following operations: collection, storage and transformation [7]. In more detail, the collection of information may include obtaining or recording of any data, storage means the holding of data on any computational system, and transformation includes alteration, combination or destruction of data.

(b) Information may be processed in any *computational system* [7]. As such, a computational system consists of one or more devices that operate automatically in response to instructions given for that purpose. Data is considered to be stored in data records (on a computational system). A data record is a collection of typically different data types abstracting the personal data of an individual in a computational system.

Privacy: In order to preserve an individual's privacy in the processing of their information, this processing must follow certain 'rules'. In the following, we outline what we mean by 'rules':

(a) A *privacy notion* abstracts and formulates a certain property or expectation of how a system should behave. In other words, a system complies with or achieves a privacy notion if it behaves in a certain way (outlined by the privacy notion). Examples include: unlinkability, pseudonymity, anonymity and unobservability [8].

(b) A *privacy model* sets out some expectations (*i.e.* a structure) how data is transformed to, or how data must 'look', in order to comply with given properties of the privacy model. The 'structure' of the resulting data may be imposed by rules (*i.e.* the rules define how to transform plain data into the 'structured' format). Moreover, a privacy model typically states guarantees of privacy which the data, by successful transformation, satisfy. Examples include: k-anonymity [408], ℓ-diversity [298], t-closeness [282] and any variants of differential privacy [159].

(c) A *privacy operation* describes a generic technique in order to transform data for the purpose of preserving an individual's privacy. Examples of privacy operations include: perturbation, generalisation, suppression, permutation and anatomisation. Fung *et al.* [187] provide a more detailed overview of privacy operations.

(d) A *privacy mechanism* expresses the mathematical abstraction which transforms sensitive input data to public 'privacy-preserved' data. Examples include: the Laplace mechanism [162], the exponential mechanism [314], and randomised response [433] for the privacy model of differential privacy [159].

(e) A *privacy axiom* is a guideline or rule which is used to enforce a certain consistency for privacy definitions and notions. Examples include: the axiom of transformation invariance and the axiom of convexity [263].

Information and Types of Disclosures: Any data that is processed in a system represents information. Information may be sensitive or non-sensitive. In the following, we define what kind of information we consider and what kind of information disclosures are in our interest:

(a) *Information* is an entity or measure that resolves uncertainty or gives an answer to a question. With *non-semantic information* we consider information that is represented and can be learned by the data itself. With *semantic information* we consider meta-information, or information that can be learned from the data. Examples of semantic information include: the length of the data, origin of the data, and structure of the data.

(b) *Information leakage* or *disclosure* describes the act of intentional and/or unintentional revealing of information.

There exist several types of disclosure methods as described in the following:

(a) With *membership disclosure* we understand that an adversary is able to determine if an individual's data is included in the dataset. This does not mean that the adversary is able to pin-point a specific tuple or attribute, but solely that the target individual's data is included.

(b) With *attribute disclosure* we understand that an adversary is able to link a sensitive attribute to a group to which an individual belongs to. As such, the adversary may not be able to precisely identify the whole tuple to which the individual belongs, but only the sensitive attribute that is leaked.

(c) With *identity disclosure* [9] we understand that an adversary is able to link an individual to a tuple in a dataset. As such, the adversary may be able to learn all attributes of this tuple.

Information Metrics: Privacy-preserving operations on information are necessary in order to protect sensitive values in the presence of adversaries. However, for further processing of the information contained in a privacy-preserving query response or an anonymised data set, the underlying information must be retained. An information metric may be used to measure the accuracy and data usefulness in any privacy-preserving process. In the literature, a privacy/utility trade-off is often mentioned when referring to information metrics. Here, we outline utility categories by which the effectiveness of a privacy operation may be measured (as a disclaimer, this list is not exhaustive and there may be other categories which fall under the wording 'utility'):

(a) Under the term *accuracy* [8] we understand measures that investigate the 'closeness'[4] of the privacy-preserving query response compared to a non privacy-preserving query response. Examples of accuracy measures include: relative

[4]By 'closeness' we mean how much, in terms of distance, the modified (*i.e.* privacy-preserved) value deviates from the original value.

and absolute error (population or sampling error, error induced by the privacy operation), the L1-norm[5] and the L2-norm[6].

(b) Under the term *efficiency* we understand measures that investigate the performance of an algorithm. Examples of efficiency measures include: time (latency and throughput), number of communication rounds, computational costs (number and size of messages) and resource costs (energy, memory, computational power).

(c) Under the term *flexibility* we understand measures that investigate the reusability or applicability of an algorithm to various application scenarios. In this context, flexibility may be measured through application to (different) synthetic datasets or by querying the algorithm using a pre-defined set of operations.

(d) Under the term *scalability* we understand measures that investigate the ability and performance of the privacy-preserving system in large-scale settings. Examples of scalability measures include (similar to efficiency measures): bandwidth, computation costs (number of participants, number of messages) and latency.

[5] http://mathworld.wolfram.com/L1-Norm.html
[6] http://mathworld.wolfram.com/L2-Norm.html

C | GUPT PRIVACY DRIVER EXTENSION

```
#!/usr/bin/env python
"""
Author:  Robin Ankele <robin.ankele@cs.ox.ac.uk>
         http://users.ox.ac.uk/~kell4062

Copyright (c) 2017, University of Oxford
All rights reserved.
"""

import logging

logger = logging.getLogger(__name__)

'''
 Helper Classes for GuptPrivacyDriver - These classes provide
    helper functions and represent the underlying structure of
    the data structure
'''

class Parameter():
  def __init__(self, name, value):
    self.name = name
    self.value = value

  def id(self):
    return self.name

  def val(self):
    return self.value

class Query():
  def __init__(self, type, budget):
    self.type = type
```

```
    self.budget = budget

  def get_query(self):
    return self.type

  def get_query_budget(self):
    return self.budget

class Value():
  def __init__(self, value, budget):
    self.value = value
    self.budget = budget

  def get_value(self):
    return self.value

  def get_value_budget(self):
    return self.budget

  def update_value_budget(self, budget):
    self.budget = budget


class User():
  def __init__(self, name, budget):
    self.name = name
    self.budget = budget
    self.values = []

  def get_user(self):
    return self.name

  def get_user_budget(self):
    return self.budget

  def update_user_budget(self, budget):
    self.budget = budget

  def add_value(self, value, budget):
    self.values.append(Value(value, budget))

  def get_value(self, value):
    for entry in self.values:
      if entry.get_value() == value:
        return entry
```

```
    raise logger.exception("Error: value not found")


class Analyst():
  def __init__(self, name, budget):
    self.name = name
    self.budget = budget

  def get_analyst(self):
    return self.name

  def get_analyst_budget(self):
    return self.budget

  def update_analyst_budget(self, budget):
    self.budget = budget

'''
 Base Class for GuptPrivacyDriver - This class serves as an
    interface for all other classes derived from the base class
    .
'''
class GuptPrivacyDriver():

  def initialize(self, epsilon_total=None, epsilon_q=None):
    """
    Must be overridden to provide execution logic for each
       record
    """
    raise logger.exception("This function should be over ridden
       ")

  def get_query_budget(self, user=None, value=None, analyst=
     None, query=None):
    """
    Must be overridden to provide execution logic for each
       record
    """
    raise logger.exception("This function should be over ridden
       ")

  def get_available_privacy_budget(self, user=None, value=None,
     analyst=None, query=None):
    """
```

```
    Must be overridden to provide execution logic for each
        record
    """
    raise logger.exception("This function should be over ridden
        ")

  def check_query_eligibility(self, user=None, value=None,
      analyst=None, query=None):
    """
    Must be overridden to provide execution logic for each
        record
    """
    raise logger.exception("This function should be over ridden
        ")

  def execute(self, user=None, value=None, analyst=None, query=
      None):
    """
    Must be overridden to provide execution logic for each
        record
    """
    raise logger.exception("This function should be over ridden
        ")

  def finalize(self):
    pass

'''
 L1PDPDriver - Level 1 Personalised Differential Privacy Driver
     Class: The level 1 personalised differential privacy class
     provides standard differential privacy, with a global
    total privacy budget and global query budget
'''
class L1PDPDriver(GuptPrivacyDriver):

  def initialize(self, epsilon_total=None, epsilon_q=None):
    self.epsilon_total = epsilon_total
    self.epsilon_q = epsilon_q

  def get_query_budget(self, user=None, value=None, analyst=
      None, query=None):
    return self.epsilon_q

  def get_available_privacy_budget(self, user=None, value=None,
      analyst=None, query=None):
```

```
    return self.epsilon_total

  def check_query_eligibility(self, user=None, value=None,
      analyst=None, query=None):
    temp_budget = self.epsilon_total
    temp_budget -= self.epsilon_q

    if temp_budget >= 0:
      return True
    return False

  def execute(self, user=None, value=None, analyst=None, query=
      None):
    self.epsilon_total -= self.epsilon_q

    if self.epsilon_total < 0:
      raise logger.exception("Privacy budget has been exceeded"
          )

'''
 L2PDPDriver - Level 2 Personalised Differential Privacy Driver
     Class: The level 2 personalised differential privacy class
     provides provenance based differential privacy. Each user
    is assigned to its own total privacy budget, while the
    query budget remains globally the same.
'''
class L2PDPDriver(GuptPrivacyDriver):

  def initialize(self, epsilon_total=None, epsilon_q=None):
    self.users = []
    self.epsilon_q = epsilon_q

  def add_entry(self, user=None, value=None, analyst=None,
      query=None):
    if user == None or user.id() == None or user.val() == None:
      raise logger.exception("Error: parameter user undefined")

    self.users.append(User(user.id(), user.val()))

  def get_available_privacy_budget(self, user=None, value=None,
      analyst=None, query=None):
    if user == None or user.id() == None:
      raise logger.exception("Error: parameter user is
          undefined")
```

```
    for entry in self.users:
      if entry.get_user() == user.id():
        return entry.get_user_budget()

    raise logger.exception("Error: user <"+ user.id() +"> not
       found")

  def get_query_budget(self, user=None, value=None, analyst=
     None, query=None):
    return self.epsilon_q

  def check_query_eligibility(self, user=None, value=None,
     analyst=None, query=None):
    if (self.get_available_privacy_budget(user=user) - self.
       epsilon_q) >= 0:
      return True
    return False

  def execute(self, user=None, value=None, analyst=None, query=
     None):
    user_budget = self.get_available_privacy_budget(user=user)
    user_budget -= self.epsilon_q

    if user_budget < 0:
      raise logger.exception("Privacy budget has been exceeded"
         )
    else:
      for entry in self.users:
        if entry.get_user() == user.id():
          entry.update_user_budget(user_budget)

'''
 L3PDPDriver - Level 3 Personalised Differential Privacy Driver
     Class: The level 3 personalised differential privacy class
     provides value based differential privacy. Each user's
    value is assigned to its own total privacy budget, while
    the query budget remains globally the same.
'''
class L3PDPDriver(GuptPrivacyDriver):

  def initialize(self, epsilon_total=None, epsilon_q=None):
    self.users = []
    self.epsilon_q = epsilon_q
```

```
def add_entry(self, user=None, value=None, analyst=None,
    query=None):
  if user == None or user.id() == None or value == None or
      value.id() == None or value.val() == None:
    raise logger.exception("Error: user or value is undefined
       ")

  for entry in self.users:
    if entry.get_user() == user.id():
      entry.add_value(value.id(), value.val())
      return

  u = User(user.id(), None)
  u.add_value(value.id(), value.val())
  self.users.append(u)

def get_query_budget(self, user=None, value=None, analyst=
    None, query=None):
  return self.epsilon_q

def get_available_privacy_budget(self, user=None, value=None,
     analyst=None, query=None):
  if user == None or user.id() == None or value == None or
      value.id() == None:
    raise logger.exception("Error: user or value is undefined
       ")

  for entry in self.users:
    if entry.get_user() == user.id():
      return entry.get_value(value.id()).get_value_budget()

  raise logger.exception("Error: user <"+ user.id() +"> not
      found")

def check_query_eligibility(self, user=None, value=None,
    analyst=None, query=None):
  if (self.get_available_privacy_budget(user=user, value=
      value) - self.epsilon_q) >= 0:
    return True
  return False

def execute(self, user=None, value=None, analyst=None, query=
    None):
  value_budget = self.get_available_privacy_budget(user=user,
      value=value)
```

```
    value_budget -= self.epsilon_q

    if value_budget < 0:
      raise logger.exception("Privacy budget has been exceeded"
          )
    else:
      for entry in self.users:
        if entry.get_user() == user.id():
          entry.get_value(value.id()).update_value_budget(
              value_budget)

'''
 L4PDPDriver - Level 4 Personalised Differential Privacy Driver
     Class: The level 4 personalised differential privacy class
     provides analyst based differential privacy. Each analyst
    who is querying the system gets a total privacy budget
    assigned. The query budget remains globally the same.
'''
class L4PDPDriver(GuptPrivacyDriver):

  def initialize(self, epsilon_total=None, epsilon_q=None):
    self.analysts = []
    self.epsilon_q = epsilon_q

  def add_entry(self, user=None, value=None, analyst=None,
      query=None):
    if analyst == None or analyst.id() == None or analyst.val()
        == None:
      raise logger.exception("Error: analyst is undefined")

    self.analysts.append(Analyst(analyst.id(), analyst.val()))

  def get_available_privacy_budget(self, user=None, value=None,
      analyst=None, query=None):
    if analyst == None or analyst.id() == None or analyst.val()
        == None:
      raise logger.exception("Error: analyst is undefined")

    for entry in self.analysts:
      if entry.get_analyst() == analyst.id():
        return entry.get_analyst_budget()

    raise logger.exception("Error: analyst <"+ analyst.id() +">
        not found")
```

```
  def get_query_budget(self, user=None, value=None, analyst=
      None, query=None):
    return self.epsilon_q

  def check_query_eligibility(self, user=None, value=None,
      analyst=None, query=None):
    if (self.get_available_privacy_budget(analyst=analyst) -
        self.epsilon_q) >= 0:
      return True
    return False

  def execute(self, user=None, value=None, analyst=None, query=
      None):
    analyst_budget = self.get_available_privacy_budget(analyst=
        analyst)
    analyst_budget -= self.epsilon_q

    if analyst_budget < 0:
      raise logger.exception("Privacy budget has been exceeded"
          )
    else:
      for entry in self.analysts:
        if entry.get_analyst() == analyst.id():
          entry.update_analyst_budget(analyst_budget)


'''
 LxPDPDriver - Level x Personalised Differential Privacy Driver
     Class: The level x personalised differential privacy class
     provides any combination of personalised differential
    privacy (defined above), with any type of query budget for
    different queries.
'''
class LxPDPDriver(GuptPrivacyDriver):

  def initialize(self, epsilon_total=None, epsilon_q=None):
    self.users = []
    self.analysts = []
    self.queries = []
    self.epsilon_total = epsilon_total
    self.epsilon_q = epsilon_q

  def add_entry(self, user=None, value=None, analyst=None,
      query=None):
    if user != None:
```

```
      if user.id() == None or user.val() == None:
        raise logger.exception("Error: user is undefined")

      add_new_user = 1
      for entry in self.users:
        if entry.get_user() == user.id() and value != None:
          if value.id() == None or value.val() == None:
            raise logger.exception("Error: value is undefined")
          entry.add_value(value.id(), value.val())
          add_new_user = 0
          break

      if add_new_user:
        if value != None:
          u = User(user.id(), user.val())
          u.add_value(value.id(), value.val())
          self.users.append(u)
        else:
          self.users.append(User(user.id(), user.val()))

    if analyst != None:
      if analyst.id() == None or analyst.val() == None:
        raise logger.exception("Error: analyst is undefined")
      self.analysts.append(Analyst(analyst.id(), analyst.val())
          )

    if query != None:
      if query.id() == None or query.val() == None:
        raise logger.exception("Error: query is undefined")
      self.queries.append(Query(query.id(), query.val()))

  def get_available_privacy_budget(self, user=None, value=None,
      analyst=None, query=None):
    budget = []

    if user != None:
      if user.id() == None:
        raise logger.exception("Error: user is undefined")
      found = 0
      for entry in self.users:
        if entry.get_user() == user.id():
          budget.append(entry.get_user_budget())
          found = 1
      if not found:
```

```
        raise logger.exception("Error: user <"+ user.id() +">
            not found")

    if value != None:
      if user == None or user.id() == None or value.id() ==
          None:
        raise logger.exception("Error: user or value is
            undefined")
      found = 0
      for entry in self.users:
        if entry.get_user() == user.id():
          budget.append(entry.get_value(value.id()).
              get_value_budget())
          found = 1
      if not found:
        raise logger.exception("Error: user <"+ user.id() +">
            or value <"+ value.id() +"> not found")

    if analyst != None:
      if analyst.id() == None:
        raise logger.exception("Error: analyst is undefined")
      found = 0
      for entry in self.analysts:
        if entry.get_analyst() == analyst.id():
          budget.append(entry.get_analyst_budget())
          found = 1
      if not found:
        raise logger.exception("Error: analyst <"+ analyst.id()
            +"> not found")

    if query != None:
      if query.name() == None:
        raise logger.exception("Error: query is undefined")
      found = 0
      for entry in self.queries:
        if entry.get_query() == query.id():
          budget.append(entry.get_query_budget())
          found = 1
      if not found:
        raise logger.exception("Error: query <"+ query.id() +">
            not found")

    return budget
```

```
def get_query_budget(self, user=None, value=None, analyst=
    None, query=None):
  if query != None:
    if query.id() == None:
      raise logger.exception("Error: query is undefined")

    for entry in self.queries:
      if entry.get_query() == query.id():
        return entry.get_query_budget()

    raise logger.exception("Error: query <"+ query.id() +">
       not found")
  else:
    return self.epsilon_q

def check_query_eligibility(self, user=None, value=None,
    analyst=None, query=None):
  budget = self.get_available_privacy_budget(user=user, value
     =value, analyst=analyst, query=query)

  epsilon_q = 0
  if query != None:
    found = 0
    for entry in self.queries:
      if entry.get_query() == query.id():
        epsilon_q = entry.get_query_budget()
        found = 1
    if not found:
      raise logger.exception("Error: query <"+ query.id() +">
          not found")
  else:
    epsilon_q = self.epsilon_q

  for entry in budget:
    if (entry - epsilon_q) < 0:
      return False
  return True

def execute(self, user=None, value=None, analyst=None, query=
    None):

  epsilon_q = 0
  if query != None:
    found = 0
    for entry in self.queries:
```

```
            if entry.get_query() == query.id():
              epsilon_q = entry.get_query_budget()
              found = 1
          if not found:
            raise logger.exception("Error: query <"+ query.id() +">
                not found")
        else:
          epsilon_q = self.epsilon_q

        if user != None:
          if value != None:
            if value.id() == None or value.val() == None or user.id
                () == None:
              raise logger.exception("Error: user or value is
                  undefined")

            for entry in self.users:
              if entry.get_user() == user.id():
                value_budget = entry.get_value(value.id()).
                    get_value_budget() - epsilon_q
                if value_budget < 0:
                  raise logger.exception("Error: value's <"+ user.
                      id() +">.<" + value.id() + "> privacy budget
                      has been exceeded")
                entry.get_value(value.id()).update_value_budget(
                    value_budget)
          else:
            for entry in self.users:
              if entry.get_user() == user.id():
                user_budget = entry.get_user_budget() - epsilon_q
                if user_budget < 0:
                  raise logger.exception("Error: user's <"+ user.id
                      () +"> privacy budget has been exceeded")
                entry.update_user_budget(user_budget)

        if analyst != None:
          for entry in self.analysts:
            analyst_budget = entry.get_analyst_budget() - epsilon_q
            if analyst_budget < 0:
              raise logger.exception("Error: analyst's <"+ analyst.
                  id() +"> privacy budget has been exceeded")
            entry.update_analyst_budget(analyst_budget)

if __name__ == '__main__':
```

```
    print >> sys.stderr, "This is a library and should not be
        executed standalone"
    sys.exit(1)
```

D | PRIVACY GAMES LIBRARY

In this chapter, we present the software implementation of our privacy games. Our software library is available as open-source implementation at:

https://github.com/robinankele/privacy-games.

The library is mainly targeted at the following demographics:

- **Privacy experts**: who are looking to develop software tools to automate the analysis of privacy-preserving applications and to generate new insights and relationships between privacy notions.
- **Software developers and system designers**: who are looking to get hands-on tools to educated and familiarise themselves with privacy tools given a common playground.

```
#!/usr/bin/env python

"""
Author:  Robin Ankele <robin.ankele@cs.ox.ac.uk>
         http://users.ox.ac.uk/~kell4062

Copyright (c) 2017, University of Oxford
All rights reserved.
"""

import sys
import os

""" Helper Classes """
class Parameter():
  def __init__(self, u_i, a_i):
    self.u_i = u_i
    self.a_i = a_i

  def identifier(self):
    return self.u_i

  def value(self):
    return self.a_i


class Party():
  def __init__(self, p_i):
    self.db = []
    self.p_i = p_i

  def add(self, u_i, a_i):
    self.db.append(Parameter(u_i, a_i))

  def get(self, u_i):
    a_ = []
    for i in range(0, len(self.db)):
      if self.db[i].identifer() == u_i:
        a_.append(self.db[i].value())
    return a_

  def validate(self, e_x, u_x):
    if len(e_x) > 1:
      e_x = e_x[0]
```

```
    for i in range(0, len(self.db)):
      if self.db[i].value() == e_x and self.db[i].identifier()
          == u_x:
        return true
    return false

  def corrupt(self):
    for i in range(0, len(self.db)):
        print("a_i: " + str(self.db[i].value()) + ", u_i: " +
            str(self.db[i].identifier()))
    return self.db


""" Basic class: Game G """
class G():
  def __init__(self):
    self.c = 0
    self.n = 0
    self.p = []
    self.__q_f = dict()
    self.__p_f = dict()

  def __finialize__(self):
    pass

  def initialise(self, n):
    self.c = 0
    self.n = n

    for i in range(0, self.n):
      self.p.append(Party(i))

  def input(self, u_i, a_i, p_i):
    self.p[p_i].add(u_i, a_i)
    self.c = self.c + 1

    if self.__q_f.get(u_i) == None:
      self.__q_f[u_i] = 1
    else:
      self.__q_f[u_i] = self.__q_f[u_i] + 1

    if self.__p_f.get(u_i) == None:
      self.__p_f[u_i] = []
    self.__p_f[u_i].append(a_i)
```

```
  def view(self, pi_phi, pi_q, op_1, op_2):
    e_ = []
    b_ = []

    for i in range(0, self.n):
      e_.append(pi_phi(self.p[i].db))
      b_.append(pi_q(self.p[i].db))

    return op_1(e_), op_2(b_)

  def corrupt(self, p_i):
    return self.p[p_i].corrupt()

  """ This should be a private function. """
  def _f(self, e_x, u_x):
    for i in range(0, self.n):
      if self.p[i].validate(e_x, u_x):
        return true
    return false

  """ This is a protected function, only accessible by
     inherited classes. """
  def _U_f(self):
    return self.__q_f.keys()

  """ This is a protected function, only accessible by
     inherited classes.  """
  def _size_U_f(self):
    return len(self.__q_f)

  """ This is a protected function, only accessible by
     inherited classes.  """
  def _Q_f(self):
    return self.__q_f.items()

  """ TThis is a protected function, only accessible by
     inherited classes.  """
  def _P_f(self):
    return self.__p_f.values()

if __name__ == '__main__':
  print >> sys.stderr, "This is a library and should not be
     executed standalone"
  sys.exit(1)
```

```
#!/usr/bin/env python

"""
Author:  Robin Ankele <robin.ankele@cs.ox.ac.uk>
         http://users.ox.ac.uk/~kell4062

Copyright (c) 2017, University of Oxford
All rights reserved.
"""

from gameBasic import G

class G_SA(G):
  def __init__(self):
    G.__init__(self)

  def __finialize__(self):
    self.G.__finialize__(self)

  def SA(self, u_x, a_x, p_x):
    G.input(self, u_x, a_x, p_x)
    e_x_, b_x = G.view(self)

    """ Perform adversarial magic to break notion """
    e_x = 0
    u_x = u_x

    return validate(e_x, u_x)

  def validate(self, e_x, u_x):
    if G._f(self, e_x, u_x):
      return true
    return false


if __name__ == '__main__':
  print >> sys.stderr, "This is a library and should not be
     executed standalone"
  sys.exit(1)
```

```
#!/usr/bin/env python

"""
Author:  Robin Ankele <robin.ankele@cs.ox.ac.uk>
         http://users.ox.ac.uk/~kell4062
```

```
Copyright (c) 2017, University of Oxford
All rights reserved.
"""

from gameBasic import G

class G_PH(G):
  def __init__(self):
    G.__init__(self)

  def __finialize__(self):
    self.G.__finialize__(self)

  def oracle_size_U_f(self):
    return G._size_U_f(self)

  def PH(self, u_x, a_x, p_x):
    G.input(self, u_x, a_x, p_x)
    e_x_, b_x = G.view(self)

    """ Perform adversarial magic to break notion """
    e_x = 0
    u_x = u_x

    return validate(e_x, u_x)

  def validate(self, e_x, u_x):
    if G._f(self, e_x, u_x):
      return true
    return false


if __name__ == '__main__':
  print >> sys.stderr, "This is a library and should not be
     executed standalone"
  sys.exit(1)
```

```
#!/usr/bin/env python

"""
Author:  Robin Ankele <robin.ankele@cs.ox.ac.uk>
         http://users.ox.ac.uk/~kell4062

Copyright (c) 2017, University of Oxford
```

```
All rights reserved.
"""

from gameBasic import G

class G_SU(G):
  def __init__(self):
    G.__init__(self)

  def __finialize__(self):
    G.__finialize__(self)

  def oracle_U_f(self):
    return G._U_f(self)

  def SU(self, u_x, a_x, p_x):
    G.input(self, u_x, a_x, p_x)
    e_x_, b_x = G.view(self)

    """ Perform adversarial magic to break notion """
    e_x = 0
    e_y = 0
    u_x = u_x

    return validate(e_x, e_y, u_x)

  def validate(self, e_x, e_y, u_x):
    if G._f(self, e_x, u_x) and G._f(self, e_y, u_x):
      return true
    return false


if __name__ == '__main__':
  print >> sys.stderr, "This is a library and should not be
      executed standalone"
  sys.exit(1)
```

```
#!/usr/bin/env python

"""
Author:  Robin Ankele <robin.ankele@cs.ox.ac.uk>
         http://users.ox.ac.uk/~kell4062

Copyright (c) 2017, University of Oxford
All rights reserved.
```

```
"""

from gameBasic import G

class G_WU(G):
  def __init__(self):
    G.__init__(self)

  def __finialize__(self):
    G.__finialize__(self)

  def oracle_U_f(self):
    return G._U_f(self)

  def oracle_Q_f(self):
    return G._Q_f(self)

  def WU(self, u_x, a_x, p_x):
    G.input(self, u_x, a_x, p_x)
    e_x_, b_x = G.view(self)

    """ Perform adversarial magic to break notion """
    e_x = 0
    e_y = 0
    u_x = u_x

    return validate(e_x, e_y, u_x)

  def validate(self, e_x, e_y, u_x):
    if G._f(self, e_x, u_x) and G._f(self, e_y, u_x):
      return true
    return false


if __name__ == '__main__':
  print >> sys.stderr, "This is a library and should not be
      executed standalone"
  sys.exit(1)
```

```
#!/usr/bin/env python

"""
Author:  Robin Ankele <robin.ankele@cs.ox.ac.uk>
         http://users.ox.ac.uk/~kell4062
```

```
Copyright (c) 2017, University of Oxford
All rights reserved.
"""

from gameBasic import G

class G_PS(G):
  def __init__(self):
    G.__init__(self)

  def __finialize__(self):
    G.__finialize__(self)

  def oracle_P_f(self):
    return G._P_f(self)

  def PS(self, u_x, a_x, p_x):
    G.input(self, u_x, a_x, p_x)
    e_x_, b_x = G.view(self)

    """ Perform adversarial magic to break notion """
    e_u_x = []
    u_x = u_x

    return validate(e_u_x, u_x)

  def validate(self, e_u_x, u_x):
    if G._f(self, e_u_x, u_x):
      return true
    return false


if __name__ == '__main__':
  print >> sys.stderr, "This is a library and should not be
      executed standalone"
  sys.exit(1)
```

```
#!/usr/bin/env python

"""
Author:  Robin Ankele <robin.ankele@cs.ox.ac.uk>
         http://users.ox.ac.uk/~kell4062

Copyright (c) 2017, University of Oxford
All rights reserved.
```

```
"""

from gameBasic import G

class G_AN(G):
  def __init__(self):
    G.__init__(self)

  def __finialize__(self):
    G.__finialize__(self)

  def oracle_U_f(self):
    return G._U_f(self)

  def oracle_P_f(self):
    return G._P_f(self)

  def AN(self, u_x, a_x, p_x):
    G.input(self, u_x, a_x, p_x)
    e_x_, b_x = G.view(self)

    """ Perform adversarial magic to break notion """
    e_u_x = []
    u_x = u_x

    return validate(e_u_x, u_x)

  def validate(self, e_u_x, u_x):
    if G._f(self, e_u_x, u_x):
      return true
    return false


if __name__ == '__main__':
  print >> sys.stderr, "This is a library and should not be
     executed standalone"
  sys.exit(1)
```

```
#!/usr/bin/env python

"""
Author:  Robin Ankele <robin.ankele@cs.ox.ac.uk>
         http://users.ox.ac.uk/~kell4062

Copyright (c) 2017, University of Oxford
```

```
All rights reserved.
"""

from gameBasic import G

class G_WA(G):
  def __init__(self):
    G.__init__(self)

  def __finialize__(self):
    G.__finialize__(self)

  def oracle_U_f(self):
    return G._U_f(self)

  def oracle_Q_f(self):
    return G._Q_f(self)

  def oracle_P_f(self):
    return G._P_f(self)

  def WA(self, u_x, a_x, p_x):
    G.input(self, u_x, a_x, p_x)
    e_x_, b_x = G.view(self)

    """ Perform adversarial magic to break notion """
    e_u_x = []
    u_x = u_x

    return validate(e_u_x, u_x)

  def validate(self, e_u_x, u_x):
    if G._f(self, e_u_x, u_x):
      return true
    return false


if __name__ == '__main__':
  print >> sys.stderr, "This is a library and should not be
      executed standalone"
  sys.exit(1)
```

```
#!/usr/bin/env python

"""
```

```
Author:  Robin Ankele <robin.ankele@cs.ox.ac.uk>
         http://users.ox.ac.uk/~kell4062

Copyright (c) 2017, University of Oxford
All rights reserved.
"""

from gameBasic import G
from random import randint

class G_UO(G):
  def __init__(self):
    G.__init__(self)
    self.b = 0

  def __finialize__(self):
    G.__finialize__(self)

  def oracle_U_f(self):
    return G._U_f(self)

  def oracle_Q_f(self):
    return G._Q_f(self)

  def oracle_P_f(self):
    return G._P_f(self)

  def view_refutable(self, pi_phi, pi_q, op_1=None, op_2=None):
    self.b = randint(0,1)

    e_ = []
    b = 0

    for i in range(0, G.n):
      if self.b == 0:
        op_1(e_, pi_phi(G.p[i]))
        op_2(b_, pi_q(G.p[i]))

    return e_, b

  def UO(self, u_x, a_x, p_x):
    G.input(self, u_x, a_x, p_x)
    e_x_, b_x = view_refutable()

    """ Perform adversarial magic to break notion """
```

```
    return validate(g)

  def validate(self, g):
    if g == self.b:
      return true
    return false


if __name__ == '__main__':
  print >> sys.stderr, "This is a library and should not be
     executed standalone"
  sys.exit(1)
```

```
#!/usr/bin/env python

"""
Author:  Robin Ankele <robin.ankele@cs.ox.ac.uk>
         http://users.ox.ac.uk/~kell4062

Copyright (c) 2017, University of Oxford
All rights reserved.
"""

import sys
import os
from random import uniform, randint
from math import log
import itertools

class PrivPrim():

  def __init__(self):
    pass

  def __finialize__(self):
    pass

  ''' privacy primitives (i.e. pi_phi) '''
  def identity(self, x):
    x_ = []
    for i in range(0, len(x)):
      x_.append(x[i].value())
    return x_
```

```
def perturbUnif(self, x, lower_bound=0, upper_bound=1):
  x_ = []
  for i in range(0, len(x)):
    x_.append(x[i].value() + uniform(lower_bound, upper_bound
       ))
  return x_

def perturbLap(self, x, lower_bound=0, upper_bound=1, mue=0,
   b=1):
  x_ = []
  for i in range(0, len(x)):
    noise_uniform = uniform(-0.5, 0.5)
    x_.append(x[i].value() + (mue - b * cmp(noise_uniform, 0)
        * log(1.0 - 2.0 * abs(noise_uniform))))
  return x_

def permutate(self, x):
  x_ = []
  for i in range(0, len(x)):
      for j in range(0, len(x[i])):
          x_.append(x[i][j])
  perm = list(itertools.permutations(x_))
  i = randint(0, len(perm))
  return perm[i]

''' data mining algorithms (i.e. pi_q) '''
def sum(self, x):
  y = 0
  for i in range(0, len(x)):
    y = y + x[i].value()
  return [y]

''' concatination algorithms (i.e. op_1, op_2) '''
''' Methods of result-degree n (vector value) '''
def none_(self, x):
  return [x]

def aggregation(self, x):
  x_ = []
  for i in range(0, len(x)):
    x_.append(x[i])
  return x_

''' Methods of result-degree 1 (single value) '''
def none(self, x):
```

```
    return x

  def average(self, x):
    y = 0
    c = 0
    for i in range(0, len(x)):
      for j in range(0, len(x[i])):
        y = y + x[i][j]
        c = c + 1
    return y / c

if __name__ == '__main__':
  print >> sys.stderr, "This is a library and should not be
      executed standalone"
  sys.exit(1)
```