

Applying the Trustworthy Remote Entity to Privacy-Preserving Multiparty Computation: Requirements and Criteria for Large-Scale Applications

Robin Ankele, Kubilay A. Küçük, Andrew Martin, Andrew Simpson

Department of Computer Science

University of Oxford

Wolfson Building, Parks Road, Oxford OX1 3QD, UK

{robin.ankele, ahmet.kucuk, andrew.martin, andrew.simpson}@cs.ox.ac.uk

Andrew Paverd

Department of Computer Science

Aalto University

P.O. Box 11000, FI-00076 Aalto, Finland

andrew.paverd@aalto.fi

Abstract—The significant improvements in technology that have been seen in recent years have resulted in a shift in the computing paradigm: from isolated computational tasks to distributed tasks executed in multi-party settings. Secure Multi-Party Computation (MPC) allows for multiple parties to jointly compute a function on their private inputs. Unfortunately, traditional MPC algorithms are inefficient in the presence of a large number of participants. Moreover, in the traditional setting, MPC is only concerned with privacy of the *input* values. However, there is often a need to preserve the privacy of individuals on the basis of the *output* of the computation. Techniques proposed by the Trusted Computing community have shown promise in the context of new secure and efficient large-scale applications. In this paper, we define and analyse several use cases related to large-scale applications of the MPC paradigm. From these use cases, we derive requirements and criteria to evaluate certain MPC protocols used for large-scale applications. Furthermore, we propose the utilisation of a Trustworthy Remote Entity and privacy-preserving algorithms to achieve *confidentiality* and *privacy* in such settings.

I. INTRODUCTION

Today’s computational tasks are significantly different from those intended at the beginning of the personal computing era. In contrast to these often isolated and private computational tasks performed by individuals, nowadays several devices, owned by one or more entities, operate together in a distributed and ubiquitous network. These operational tasks are no longer isolated and conducted privately, but, rather, involve the joint execution of computational tasks on the private information of multiple parties.

The secure computation of a function in such a setting is referred to in the literature as secure multi-party computation (MPC) [1]. It is one of the most general and important problems in cryptography. The secure multi-party computation problem is defined by a set of parties P_1, \dots, P_n , each with their private input x_1, \dots, x_n , jointly computing a function $f(x_1, \dots, x_n) = y$. To achieve security in the MPC setting, the properties of (*input*) *privacy*, *correctness*, *independence of inputs*, *fairness* and *guaranteed output delivery* as defined in [1] must be fulfilled by any protocol instantiation. Yet, entities participating in the MPC protocol might be mutually untrusted. Lindell *et al.* [1] elaborate on the adversarial

model of *static* and *adaptive* adversaries that demonstrate *semi-honest* (i.e. *passive*), *covert* or *malicious* (i.e. *active*) behaviour.

The simplest solution to this problem is to share the private data with a mutually trusted party — with that party being trusted to compute the function. However, such a trust relationship can not always be established and so early results from the 1980s have shown the feasibility of replacing the trusted party with a cryptographic protocol to achieve the same level of security. Theoretical research in the 1980s and 1990s gave rise to the results of Yao [2], the GMW [3] protocol by Goldreich, Micali and Wigderson, the BMR [4] protocol by Beaver, Micali and Rogaway, and the BGW [5] protocol by Ben-Or, Goldwasser and Wigderson. All of these protocols utilise garbled circuits: the Yao, GMW and BMR protocols are based on Boolean circuits; the BGW protocol is based on arithmetic circuits. Moreover, the underlying cryptographic primitive of Yao and GMW is *oblivious transfer* [6]; BMR and BGW make use of *secret sharing* [7], [8].

All of the above protocols elaborate on a general approach to MPC, whereas there has been limited research on function-specific MPC. This results from insignificant performance improvements of function specific protocols and the more general applicability of a non-function specific protocol to real-world applications. However, through recent developments and tremendous improvements in technology, there has been an increase in the development of practical MPC protocols. This includes the results on fully homomorphic encryption (FHE) [9] by Gentry and some combinations and real-world implementations of the above mentioned protocols — VIFF [10], TASTY [11], Sharemind [12] and Fairplay [13].

Unfortunately, most of these implementations are far from practical. The most common shortcomings of recently proposed MPC implementations are: a huge communication overhead between the involved parties [14]; enormous key sizes [15], [16]; and the extensive use of cryptographic operations in case of malicious entities [17]. Solving the MPC problem efficiently is one of the key goals of cryptography.

Nevertheless, to be applicable to real-world applications, MPC faces an additional problem. The above mentioned protocols are only concerned with the problem of *input privacy*. However, the privacy-preservation of the output function computation itself (*i.e.* the result of $f()$), which we refer to as *output privacy*, is not considered. Hence, if the output of the function leaks information on the input, the whole MPC protocol fails.

Consider the following example as adapted from [1]. Two or more parties $P_1 \dots P_n$ want to compute the average of their earnings. However, consider $n - 1$ colluding parties. Without any additional checks and the knowledge of the colluding parties, the output of the *average* computation leaks the *input* of party P_n . In the case of a two-party computation, each of the two parties can easily learn the opposite input with the knowledge of their own earnings and the average earnings by simply reversing the operations.

In this paper, we explore this issue in the context of emerging widespread applications. In more detail, we include applications from the traditional MPC setting, but focus on bigger classes of problems — large-scale applications operating with big data, complex operations or a large number of participants. As a first step in this direction, we give consideration to use cases, and establish requirements and criteria for such applications.

In this context, Trusted Computing (TC) [18] has emerged from the need that a computing system should be trusted to consistently behave in an expected way enforced by hardware and software. TC provides techniques that guarantee secure computation in a potentially hostile environment. These comprise attestation, isolation of memory regions, and secure storage of data values. Moreover, TC ensures that each user is running the correct software (*i.e.* protocol), by measuring and attesting the software stack. Currently available applications include cloud computing [19], digital rights management [20] and verification of outsourced computation in grid computing [21].

In order to overcome the shortcomings of current MPC implementations we propose to apply the Trustworthy Remote Entity (TRE) [22] to the MPC setting. In contrast to the concept of a trusted third party in the *ideal world* model of the MPC setting, a TRE uses concepts of the TC domain in order to prove its trustworthy behaviour to other parties. Such a TRE can be established using a trusted platform module (TPM) introduced by the TCG¹, or by using trusted execution environments (TEE) such as ARM Trustzone^{®2} [23] or Intel[®] SGX [24]. Moreover, to achieve *output privacy* in this setting, we propose to deploy privacy-preserving algorithms on the TRE. This concept is applicable to a wide range of MPC applications, including electronic

voting schemes, auctions, genomics, intrusion detection and privacy-preserving data mining.

The remainder of this paper is organised as follows. In Section II, we motivate the approach of using the TRE for privacy-preserving multi-party computation. Section III provides some background on the TRE and Intel[®] SGX. In Section IV, we present and analyse selected use cases for the application of privacy-preserving multi-party computation. In Section V, we derive requirements and criteria from the previous analysis for large-scale privacy-preserving multi-party applications. In Section VI, we summarise the contribution of this paper and, importantly, outline our plans for future work in this area.

II. MOTIVATION

A wide range of new applications³ for distributed, ubiquitous and secure multi-party computation has evolved from the concept of ‘big data’ and the tremendous improvements and developments in technology for the Internet of Things (IoT). These applications handle sensitive inputs (*i.e.* personally-identifying values), which arise from primary artefacts or patterns and correlations in data that may be derived in an unexpected or previously infeasible way in the presence of potential adversaries. Hence, new confidentiality and privacy challenges arise. In order to address these challenges, there is clearly a need for practical MPC solutions that operate in a secure and privacy-preserving way.

Unfortunately, currently proposed MPC solutions suffer from inefficiency with regard to complex operations and/or large-scale applications. Examples include the following.

Communication Overhead. Many proposed protocols have large polynomials as their computational and communication complexity (*e.g.* $\mathcal{O}(n^8)$ for an implementation of the BGW protocol [5], [14], where n represents the number of participating parties). Moreover, most protocols are dependent on factors [27] such as: the circuit size s ; the number of participants n ; the security parameter λ ; or the rounds of communication d . In order to establish feasible protocols for large-scale implementations, protocols should be independent of these factors. Therefore, Damgård and Nielsen [28] proposed MPC protocols that reduce to linear complexity of $\mathcal{O}(n)$ for *passive* security, and quadratic complexity $\mathcal{O}(n^2)$ in the case of *active* security. Additionally, to solve communication complexity in large-scale applications one must consider the concepts of *communication locality* [29], *load balancing* [29] or building so-called *quorums* [30].

Enormous Key Sizes. To achieve a moderate level of security (*i.e.* adversary attack complexity) in a fully homomorphic encryption based MPC scheme, there is a need for enormous key sizes (*e.g.* between 800 MB and 2.3 GB for a public key and a security level $\lambda = 72$ bits [15], [16]). Another potential problem is that each participant in the

¹Trusted Computing Group, <http://trustedcomputinggroup.org>

²Using ARM Trustzone[®] additional attestation protocols are required in order to construct a TRE.

³See [25] for a discussion of challenges and applications in ‘big data’ and [26] for a discussion of open problems in multi-party computation.

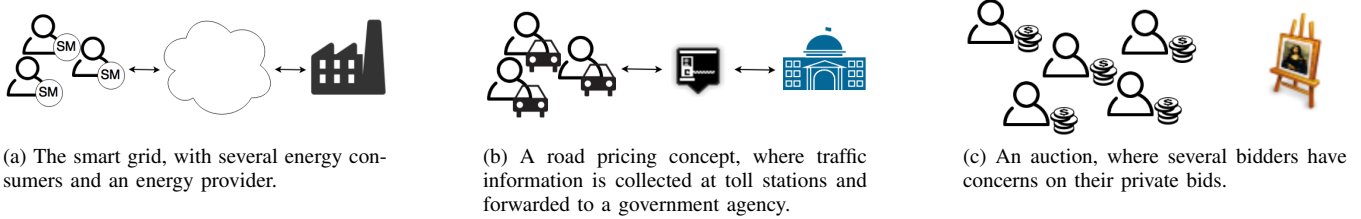


Figure 1: Illustration of MPC examples from Section IV.

computation needs to share the same key to perform the homomorphic operations, which again leads to a communication overhead during the key exchange.

Use of Expensive Operations. Guaranteeing privacy for an MPC protocol in the malicious setting is typically harder and more expensive. Hence, it involves the use of complex operations and results in an overhead in communication (*e.g.* the cut-and-choose technique [17]). Moreover, depending on the number of corrupted parties, MPC properties such as fairness or guaranteed output delivery can not be obtained [1].

Moreover, current MPC protocol solutions are only concerned with *input privacy*, but do not protect against any leaks from the function's output. However, this type of leakage can result in failure of the privacy requirement of the whole MPC protocol. Hence, there is a clear need to implement privacy-preserving methods for the function output.

Research on Trusted Computing (TC) techniques has given rise to the establishment of secure and trustworthy applications in various computing fields (*e.g.* large-scale computing, cloud computing, and multi-party settings). Contrary to the typically more expensive cryptographic approaches, TC offers often a more general-purpose and efficient approach with still reasonable security guarantees. Especially for large-scale applications, the trade-off between the strong mathematical guarantees of the cryptographic approach to the efficiency of TC must be considered.

In this context, and in order to improve upon the above shortcomings of MPC, we propose a different approach. Our model takes elements from the TC domain and combines them with privacy algorithms from the privacy-preserving data mining domain to achieve *input* and *output* preservation of MPC applications. In particular, we use the concept of the Trustworthy Remote Entity of Paverd [22].

A TRE is developed using trusted execution environments, where in particular Intel[®] SGX instructions seem to be a good fit. The advantages of SGX include a reduced trusted computing stack, isolation of program code and data from other applications, secure storage, and the possibility of local and remote attestation to prove its trustworthy behaviour to other parties.

Additionally, the combination of the TRE with privacy-preserving algorithms such as differential privacy [31] can help to prevent algorithmic re-identification of data artefacts

and patterns from the processed output function. Furthermore, application users are able to specify a privacy-budget to manifest their level of trust in the computation.

III. BACKGROUND

In the following, we present some background information on secure multi-party computation, the Trustworthy Remote Entity and Intel[®] Software Guard Extensions.

A. Secure Multiparty Computation

The secure multi-party computation problem is defined as follows. A set of parties P_1, \dots, P_n , each with their private input x_1, \dots, x_n , jointly compute a function $f(x_1, \dots, x_n) = y$. Here, $f()$ is not restricted and can be any function whatsoever. Moreover, to achieve security in the MPC setting, the properties of (*input*) *privacy*, *correctness*, *independence of inputs*, *fairness* and *guaranteed output delivery* as defined in [1] must be fulfilled by any protocol instantiation. Figure 1 illustrates some MPC examples, which are discussed in Section IV.

B. Trustworthy Remote Entity

Paverd introduced the concept of a Trustworthy Remote Entity [22]. His existing TRE prototype (24k lines of code, bare metal application using TXT⁴ and the TPM) addresses the smart grid instance [32]. Contrary to a trusted third party, users are not required to blindly trust the operations of a Trustworthy Remote Entity. Instead, the TRE makes use of techniques from the TC domain (*e.g.* remote attestation) to ensure its technical guarantees of trustworthiness. Figure 2 illustrates the features of a TRE — isolation of code and data, secure storage of data, and remote attestation. We note here that a TRE can prove its trustworthy behaviour to either communication partners or another TRE. Additionally, a TRE is capable of privacy-preserving bi-directional communication. The TRE implementation of Paverd utilises a final state attestation protocol, which improves the performance of the attestation by a factor of 16 (or approximately 1500%) [22], compared to an attestation based on a TPM v1.2. Using a SGX-based TRE implementation, it is likely that the attestation performance is significantly higher, since all operations are carried out on the main CPU. Moreover,

⁴Intel[®] Trusted Execution Technology

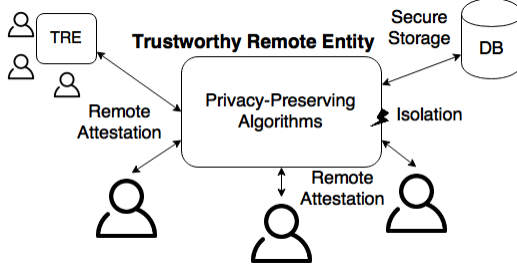


Figure 2: Features of a Trustworthy Remote Entity: remote attestation, isolation, and secure storage.

using Intel[®] SGX instructions, we can ensure to minimise the Trusted Computing Base in order to achieve effective and low-cost results.

C. Intel[®] SGX

Intel[®] Software Guard Extension [24], [33], [34] allows the allocation of private memory locations within user-space code to protect against other potentially malicious processes at the same or a higher level such as kernel-space drivers, the operating system or virtual machine manager. Such an application can contain several trusted parts (so called *enclaves*) and untrusted parts. Moreover, SGX ensures hardware-enforced *isolation* of these trusted memory pages (*i.e.* code and data pages) and supports *secure storage* of these on the hard disk. (See Figure 3 for more details on the above concepts). Furthermore, an application using enclaves can perform *local* and *remote attestation* to other enclaves. Applications deploying SGX can therefore effectively minimise the Trusted Computing Base, which comes in very handy when building large-scale applications. Additionally, Hoekstra *et al.* [35] present some applications using SGX technology.

IV. USE CASES

We have defined several use cases for large-scale applications in the multi-party setting. We split these use cases into two parts, on the basis of their underlying computational model — either the client-server model (*i.e.* many-to-one) or the MPC model (*i.e.* many-to-many). We note that a comprehensive survey of all applicable MPC use cases would be impossible; rather, we focus on considering a representative sample of use cases to motivate and validate our research. Additionally, due to space limitations, we have chosen to characterise each use case briefly.

A. Smart Grid

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 in a more efficient and economical way. We have identified the following three information flows of a modern smart grid. The first two follow the client-server model, whereas the third follows the MPC model.

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 [36] a potential malicious energy provider can detect patterns and artefacts 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 or dynamic billing. While in the time-of-use scheme, the price varies 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 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

B. Road Pricing

New tolling and road pricing schemes need to capture and analyse a high load of privacy-sensitive values during certain times of the day. 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.

PROBLEM 4. (Traffic Monitoring) Traffic authorities have an increasing interest in gathering traffic information and

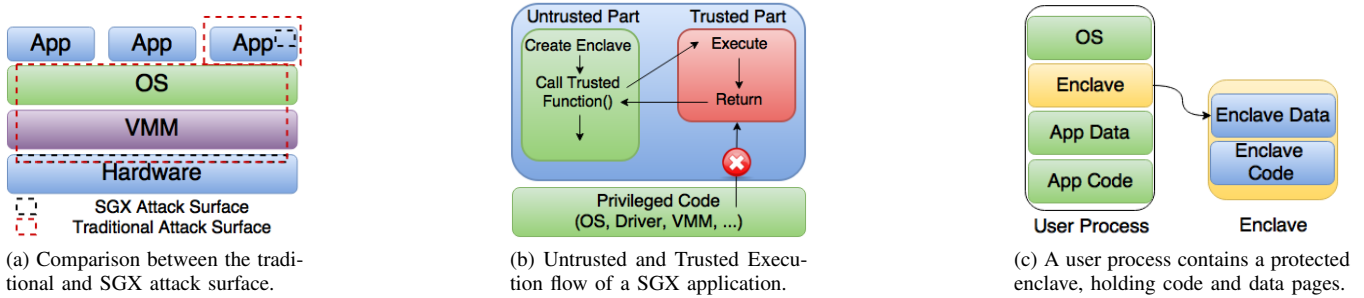


Figure 3: Features of the Intel® SGX technology.

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.* license 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 in forms of 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 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

C. Privacy-Preserving Data Mining

Privacy-preserving data mining 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 TC 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 mining following the MPC model.

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); 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 persons⁵. 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

PROBLEM 7. (Intrusion detection) In the intrusion detection use case we represent three different scenarios. In the first scenario, 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, employees of a company are monitored to detect if they leak any sensitive-information to outsiders (*e.g.* competitive companies). Thereby, the be-

⁵For 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.

haviour 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

D. Auctions

In an auction, participants 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.

PROBLEM 8. (Auctions) In order to select the winner of an auction, the auctioneer must learn the inputs of all participating parties. From the public announcement of the bid, other parties learn certain information about each of the participants. Moreover, often even the detection of participation in an auction can reveal sensitive information. Therefore, a system modelling this scenario has to produce the correct output (*i.e.* the winner of the auction), obfuscate the participation, and protect the private inputs of each participant.

Analysis. (Sensitive values) privacy of bids, anonymity of bidder

E. Elections

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 TC and MPC, the trusted third party can be dropped.

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

F. Set Intersections

The problem of set intersections is one of the classical MPC problems⁶. The problem is similarly stated to the privacy-preserving data mining problem. In a 'big-data' environment, set intersection applications can benefit from the TC and MPC approaches.

PROBLEM 10. (Intersections) The problem states two or more parties that want to determine the amount of intersection between their private databases. Since the databases contain privacy-sensitive values, it is not possible to just share the data and perform analysis on the joint database. A solution must reveal the amount of how much the sets overlap (*i.e.* 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

V. REQUIREMENTS AND CRITERIA

In the following, we present the requirements and criteria for large-scale applications derived from the analysis of the use cases presented in Section IV. We split this analysis into two sections, whereby we discuss requirements for the design of prototypes for privacy-preserving multi-party large-scale applications in Section V-A and present criteria for the evaluation of these prototypes in Section V-B.

A. Requirements for the Design of Large-Scale Prototypes

Each of the above stated use cases handles privacy-sensitive data values of several data owners. In order to collaboratively evaluate a function on these data values, the mutually distrustful parties have to find a way to share those values. However, data owners are not willing to disclose their data in a plain form, as it might leak any information on their privacy-sensitive data values. In addition to the protection of the input values, there is also the requirement that the computational function itself does not leak any information of the input values. Therefore, the first requirement derives from this problem.

Requirement 1. We require *privacy-preservation* of the *input* and *output*. We refer to the former as *input privacy* and to the latter as *output privacy*.

A system that fulfills the *input privacy* requirement handles the input values in a confidential and integrity-preserving manner and does not leak any information on the private inputs. Moreover, a system fulfilling the *output privacy* requirement does not leak any information on the private inputs from the function evaluation.

Often even 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

⁶See Yao's Millionaires Problem [2].

election) is desired. As a second requirement we state the following.

Requirement 2. We require *anonymity* of parties participating in the protocol. However, we also require *non-repudiation* of committed data 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. Moreover, this requirement must hold even in the case of a majority of malicious parties (preferably, such a system is independent of the number of malicious parties).

Requirement 3. We require *correctness* of the function evaluation, in the presence of malicious parties.

Following the *correctness* requirement, we further harden such a system by requiring guaranteed output delivery of the result of the computational function evaluation.

Requirement 4. We require *guaranteed output delivery* of the function evaluation to all trustworthy and malicious parties.

To refine the requirement of *output privacy*, we introduce the approach of a *privacy budget* for the level of private information leaked by the output of the computational function. This ‘budget’ should be user-configurable and further improve on the accuracy of certain function evaluations⁷.

Requirement 5. The parties should be able to maintain a *privacy budget* devoting the level of trust in the application.

In order to be deployable for real-world applications, such a system must be computable in a reasonable time. This is, however, dependent on the computational complexity of the function, the number of participants, and several other factors. Nevertheless, it represents a major factor in the selection of applications. Section V-B elaborates on criteria for efficient and scalable implementations.

Requirement 6. We require an *efficient* and *scalable* implementation of the privacy-preserving multi-party protocol.

B. Evaluation Criteria of Large-Scale Prototypes

In the following, we discuss several metrics and criteria for the evaluation of a large-scale prototype. We split them into four groups — *computational models*, *general criteria*, *large-scale criteria* and *additional metrics*.

1) *Computational Models*: First, we introduce the communication model. The *synchronous* model defines an upper bound, known to all participating parties, on the time it takes for a message to reach the intended receiver. The *asynchronous* model, on the contrary, has no such limitations. However, in a real-world protocol, if a certain time threshold is exceeded, the computation has to either fail or to continue, whereas the *correctness* or *guaranteed output delivery* requirement suffers from such a model. In addition

to the above stated models, one has to consider if there is a need for *uni* or *bi-directional* communication.

Secondly, we introduce the adversarial model. We consider an adversary as *semi-honest* if it tries to learn information of other parties’ input values, but strictly follows the given protocol. Furthermore, we consider an adversary as *malicious* if it tries to learn information of other parties’ input values, but is not limited in how it does that. Therefore, it can arbitrarily deviate from the protocol and block, modify or replay any message or synthesise falsified messages. Additionally, we consider an adversary as *covert* if it only deviates from the protocol when there is a high likelihood that it stays hidden. An adversary is assumed to be either *static* or *adaptive*. The former is limited to select the set of compromised parties prior to the protocol execution, whereas the latter does not have such limitations, and can *adaptively* compromise parties during protocol runtime. Consequently, we define the metric of *fault tolerance*, which represents the number of nodes an adversary can take over without compromising *correctness* or *guaranteed output delivery*.

2) *General Criteria*: In order to measure the effectiveness of an MPC protocol from a generic and high level viewpoint the following metrics can be examined. We note that these factors are typically stated in terms of computational complexity. First, this includes *communication costs*, such as the number and size of messages (m, ℓ) sent throughout protocol execution, and the number of participants n . Secondly, it includes *computational costs*, such as the time and memory complexity ($\mathcal{O}(n), \mathcal{O}(m \cdot \ell)$) of a certain protocol. Thirdly, the *latency* metric defines the number of communication rounds d until the protocol terminates.

3) *Large-Scale Criteria*: For the evaluation of large-scale protocol instantiations the following metrics and criteria can be considered. First, this involves the usage of *parallelization* techniques to handle more parties simultaneously. Secondly, protocol instantiations can utilise *offline* and *online* phases. Thereby, precomputation of certain values, attestation or key exchanges can be handled in *offline* phases, while the message exchange is handled in the *online* phase of the protocol. Thirdly, the inclusion of *hardware-assisted design* is another criterion, whereby instruction set extensions such as SIMD⁸, SHA-EXT⁹ and SGX (see Section III-C) are a good fit. Fourthly, this involves *communication locality* [29], which states the number of nodes each party in the protocol has to communicate with. Finally, this includes *load balancing* [29], which defines the balance of computational tasks each party in the protocol has to perform.

4) *Additional Metrics*: The following metrics can be seen as more generic and additional to the design of efficient long-term protocols that are not use case specific and therefore applicable to most MPC applications. Such metrics include

⁷Alice, less concerned about the privacy-leakage of her inputs, can further improve the accuracy of the output of the computational function by defining a lower privacy budget value.

⁸Single Instruction Multiple Data (e.g. Intel[®] SSE, AVX instructions)

⁹Intel[®] SHA extensions

the *reusability* of a protocol or parts of a protocol. This especially play an important role in the design of general or function-specific protocols. Moreover, when designing a protocol, it is important to keep the *interoperability* and a *minimal design* approach of the protocol in mind. This ensures the reusability of building blocks of the protocol and keeps it as lightweight as possible. Above all, the universal composability framework [37] of Canetti defines a general-purpose model for the analysis of cryptographic protocols. Thereby, protocols remain provable secure even when arbitrarily combined with other protocol instantiations. Finally, this includes the *privacy impact* of certain functions. Typically, *privacy* is context-dependent and dependent on the quasi identifying information that an adversary already knows. However, if this information can be approximated, the *privacy impact* of the function can be used as a metric.

VI. CONCLUSION

Current implementations and protocols for practical MPC suffer from limitations and shortcomings, especially in a large-scale setting. These comprise a communication overhead, enormous key sizes in the implementation of fully homomorphic approaches, and the use of expensive operations when confronted with malicious parties. In this paper, we have proposed several use cases for the application in a large-scale and multi-party setting. These use cases handle sensitive data values of multiple parties and require privacy-preservation of each parties' input and privacy-preservation of the result of the collaborative function evaluation. We have performed a restrictive analysis of those use cases, and derived requirements and criteria for the design and evaluation of privacy-preserving large-scale protocols in the multi-party setting.

Using these results as a foundation, we can now outline our planned research work in this direction. The research is split into two strands. The first strand involves working on the development of the TRE for data confidentiality management; the second strand involves working on privacy-preserving analytics. The former is using Intel® SGX instructions and techniques of the TC domain to build a TRE. This work is also focusing on architectural design decisions for large-scale deployment. The latter is focusing on the development of privacy-preserving algorithms running on the TRE. Additionally, it is concerned with the development of metrics to measure the privacy impact of functions and the introduction of the concept of a privacy budget in the MPC setting. Therefore, each work strand is implementing a prototype, utilising privacy by design and the architectural design of a large-scale deployment of the TRE.

Generally, the TRE makes an interesting assurance target: in Paverd's work, he proves that his prototype is privacy-preserving. The extension of that project may prove other properties of a TRE too. In some ways such work narrows the gap between crypto-based MPC and our TRE approach.

Overall, this work is based on the development of large-scale applications in the multi-party setting, where we aim to achieve *input* and *output privacy*. It is hoped that this work will significantly improve the real-world applicability of such applications, and will lead to the first practical results in this setting.

ACKNOWLEDGMENT

The authors are supported by a research grant from the Intel Corporation. We are also grateful for the valuable comments by Kirk Rockett on the genomics use case.

REFERENCES

- [1] Y. Lindell and B. Pinkas, "Secure Multiparty Computation for Privacy-Preserving Data Mining," *Journ. of Privacy and Confidentiality*, vol. 1, no. 1, pp. 59–98, 2009.
- [2] A. Yao, "Protocols for Secure Computations," in *Proc. of the 23rd Symp. on Found. of C.S.*, ser. SFCs 1982. IEEE Comp. Society, 1982, pp. 160–164.
- [3] O. Goldreich, S. Micali, and A. Wigderson, "How to Play ANY Mental Game," in *Proc. of the 19th ACM Symp. on Theory of Comp.*, ser. STOC 1987. ACM, 1987, pp. 218–229.
- [4] D. Beaver, S. Micali, and P. Rogaway, "The Round Complexity of Secure Protocols," in *Proc. of the 22nd ACM Symp. on Theory of Comp.*, ser. STOC 1990. ACM, 1990, pp. 503–513.
- [5] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness Theorems for Non-cryptographic Fault-tolerant Distributed Computation," in *Proc. of the 20th ACM Symp. on Theory of Comp.*, ser. STOC 1988. ACM, 1988, pp. 1–10.
- [6] M. Rabin, "How to Exchange Secrets by Oblivious Transfer," Aiken Comp. Lab., Harvard Univ., Tech. Rep. 81, 1981.
- [7] A. Shamir, "How to Share a Secret," *Comm. of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.
- [8] G. Blakley, "Safeguarding Cryptographic Keys," in *Proc. of the AFIPS National Comp. Conf.*, vol. 48, 1979, pp. 313–317.
- [9] C. Gentry, "A Fully Homomorphic Encryption Scheme," Ph.D. dissertation, Stanford Univ., 2009.
- [10] I. Damgård, M. Geisler, M. Krøigaard, and J. Nielsen, "Asynchronous Multiparty Computation: Theory and Implementation," in *Proc. of the 12th Int. Conf. on Practice and Theory in Public Key Cryptography - PKC 2009*, ser. Lecture Notes in C.S., S. Jarecki and G. Tsudik, Eds., vol. 5443. Springer, 2009, pp. 160–179.
- [11] W. Henecka, S. Kögl, A. Sadeghi, T. Schneider, and I. Wehrenberg, "TASTY: Tool for Automating Secure Two-party Computations," in *Proc. of the 17th ACM Conf. on Comp. and Comm. Security*, ser. CCS 2010. ACM, 2010, pp. 451–462.
- [12] D. Bogdanov, S. Laur, and J. Willemson, "Sharemind: A Framework for Fast Privacy-Preserving Computations," in *Proc. of the 13th European Symp. on Research in Comp. Security - ESORICS 2008*, ser. Lecture Notes in C.S., S. Jajodia and J. Lopez, Eds., vol. 5283. Springer, 2008, pp. 192–206.

- [13] A. Ben-David, N. Nisan, and B. Pinkas, "FairplayMP: A System for Secure Multi-party Computation," in *Proc. of the 15th ACM Conf. on Comp. and Comm. Security*, ser. CCS 2008. ACM, 2008, pp. 257–266.
- [14] M. Hirt, U. Maurer, and B. Przydatek, "Efficient Secure Multi-party Computation," in *Proc. of the 6th Int. Conf. on the Theory and Application of Cryptology and Information Security - ASIACRYPT 2000*, ser. Lecture Notes in C.S., T. Okamoto, Ed., vol. 1976. Springer, 2000, pp. 143–161.
- [15] J. Coron, A. Mandal, D. Naccache, and M. Tibouchi, "Fully Homomorphic Encryption over the Integers with Shorter Public Keys," in *Proc. of the 31st Cryptology Conf. - CRYPTO 2011*, ser. Lecture Notes in C.S., P. Rogaway, Ed., vol. 6841. Springer, 2011, pp. 487–504.
- [16] C. Gentry and S. Halevi, "Implementing Gentry's Fully-Homomorphic Encryption Scheme," in *Proc. of the 30th Int. Conf. on the Theory and Applications of Cryptographic Techniques - EUROCRYPT 2011*, ser. Lecture Notes in C.S., K. Paterson, Ed., vol. 6632. Springer, 2011, pp. 129–148.
- [17] Y. Lindell and B. Pinkas, "Secure Two-Party Computation via Cut-and-Choose Oblivious Transfer," in *Proc. of the 8th Theory of Cryptography Conf. - TCC 2011*, ser. Lecture Notes in C.S., Y. Ishai, Ed., vol. 6597. Springer, 2011, pp. 329–346.
- [18] A. Martin, "The Ten Page Introduction to Trusted Computing," Univ. of Oxford, Comp. Lab., Tech. Rep. RR-08-11, 2008.
- [19] N. Santos, K. Gummadi, and R. Rodrigues, "Towards Trusted Cloud Computing," in *Proc. of the Conf. on Hot Topics in Cloud Comp.*, ser. HotCloud 2009. USENIX Ass., 2009.
- [20] Y. Zheng, D. He, H. Wang, and X. Tang, "Secure DRM Scheme for Future Mobile Networks based on Trusted Mobile Platform," in *Proc. of the Conf. on Wireless Comm., Networking and Mobile Comp.*, ser. WCNM 2005. IEEE Comp. Society, 2005, pp. 1164–1167.
- [21] W. Mao, A. Martin, H. Jin, and H. Zhang, "Innovations for Grid Security from Trusted Computing," in *Revised Selected Papers of the 14th Int. W.S. on Secure Protocols*, ser. Lecture Notes in C.S., B. Christianson, B. Crispo, J. Malcolm, and M. Roe, Eds., vol. 5087. Springer, 2006, pp. 132–149.
- [22] A. Paverd, "Enhancing Communication Privacy Using Trustworthy Remote Entities," DPhil dissertation, Univ. of Oxford, 2016.
- [23] *ARM Security Technology - Building a Secure System using TrustZone Technology*, http://infocenter.arm.com/help/topic/com.arm.doc.prtd29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf, ARM, 2009.
- [24] *Intel Software Guard Extensions Programming Reference*, <https://software.intel.com/sites/default/files/managed/48/88/329298-002.pdf>, Intel, 2014.
- [25] A. Cuzzocrea, "Privacy and Security of Big Data: Current Challenges and Future Research Perspectives," in *Proc. of the 1st Int. W.S. on Privacy and Security of Big Data*, ser. PSBD 2014. ACM, 2014, pp. 45–47.
- [26] W. Du and M. Atallah, "Secure Multi-party Computation Problems and Their Applications: A Review and Open Problems," in *Proc. of the W.S. on New Security Paradigms*, ser. NSPW 2001. ACM, 2001, pp. 13–22.
- [27] I. Damgård, Y. Ishai, and M. Krøigaard, "Perfectly Secure Multiparty Computation and the Computational Overhead of Cryptography," in *Proc. of the 29th Int. Conf. on the Theory and Applications of Cryptographic Techniques - EUROCRYPT 2010*, ser. Lecture Notes in C.S., H. Gilbert, Ed., vol. 6110. Springer, 2010, pp. 445–465.
- [28] I. Damgård and J. Nielsen, "Scalable and Unconditionally Secure Multiparty Computation," in *Proc. of the 27th Int. Cryptology Conf. - CRYPTO 2007*, ser. Lecture Notes in C.S., A. Menezes, Ed., vol. 4622. Springer, 2007, pp. 572–590.
- [29] E. Boyle, K. Chung, and R. Pass, "Large-Scale Secure Computation: Multi-party Computation for (Parallel) RAM Programs," in *Proc. of the 35th Cryptology Conf. - CRYPTO 2015*, ser. Lecture Notes in C.S., R. Gennaro and M. Robshaw, Eds., vol. 9216. Springer, 2015, pp. 742–762.
- [30] J. Saia and M. Zamani, "Recent Results in Scalable Multi-Party Computation," in *Proc. of the 41st Int. Conf. on Current Trends in Theory and Practice of C.S. - SOFSEM 2015*, ser. Lecture Notes in C.S., G. Italiano, T. Margaria-Steffen, J. Pokorný, J. Quisquater, and R. Wattenhofer, Eds., vol. 8939. Springer, 2015, pp. 24–44.
- [31] C. Dwork, "Differential Privacy," in *Proc. of the 33rd Int. Colloquium - ICALP 2006*, ser. Lecture Notes in C.S., M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, Eds., vol. 4052. Springer, 2006, pp. 1–12.
- [32] A. Paverd, A. Martin, and I. Brown, "Security and Privacy in Smart Grid Demand Response Systems," in *Proc. of the 2nd Int. W.S. - SmartGridSec 2014*, ser. Lecture Notes in C.S., J. Cuellar, Ed., vol. 8448. Springer, 2014, pp. 1–15.
- [33] V. Costan and S. Devadas, "Intel SGX Explained," Cryptology ePrint Archive, Report 2016/086, 2016.
- [34] F. McKeen, I. Alexandrovich, A. Berenzon, C. Rozas, H. Shafi, V. Shanbhogue, and U. Savagaonkar, "Innovative Instructions and Software Model for Isolated Execution," in *Proc. of the 2nd Int. W.S. on Hardware and Architectural Support for Security and Privacy*, ser. HASP 2013. ACM, 2013.
- [35] M. Hoekstra, R. Lal, P. Pappachan, V. Phegade, and J. Del Cuvillo, "Using Innovative Instructions to Create Trustworthy Software Solutions," in *Proc. of the 2nd Int. W.S. on Hardware and Architectural Support for Security and Privacy*, ser. HASP 2013. ACM, 2013.
- [36] G. Hart, "Nonintrusive Appliance Load Monitoring," *Proc. of the IEEE*, vol. 80, no. 12, pp. 1870–1891, 1992.
- [37] R. Canetti, "Universally Composable Security: A New Paradigm for Cryptographic Protocols," in *Proc. of the 42nd IEEE Symp. on Found. of C.S.*, ser. FOCS 2001. IEEE Comp. Society, 2001, pp. 136–145.