Identification and Mitigation of Attacks in Trust-Based Distributed Communication Networks

Tanmayee Deshprabhu

Supervisor: Professor Justin P. Coon

.

University of Oxford

Department of Engineering Science

.

A thesis presented for the degree of

Doctor of Philosophy

Hilary Term 2024

Identification and Mitigation of Attacks in Trust-Based Distributed Communication Networks

Name: Tanmayee Deshprabhu

College: Oriel

Degree: Doctor of Philosophy in Engineering Science

Supervisor: Professor Justin P. Coon

Term and Year of Submission: Hilary 2024

Network security is a vital component of modern wireless communications, particularly as a result of the value of data being shared within those networks. Trust inference has provided an intuitive and accessible way for distributed networks to protect themselves in lieu of a central authority, such as a basestation. However, the trust metrics themselves can be manipulated by malicious parties in order to disrupt the network.

This thesis examines the vulnerabilities that exist within distributed communication networks that rely on trust inference for security. Chapter 3 studies the landscape of trust network vulnerabilities and presents a design for a unified framework of trust network attacks from an observer's perspective, based on the comparison and collation of pre-attack symptoms. This is then used as the basis for a novel contextual characterisation method for the classification of nodes' behaviour into possible attack scenarios, which is implemented using a support vector machine.

Chapter 4 presents the design and analysis of a trust-based data management and aggregation protocol for facilitating secure self-management of distributed network nodes. This is accompanied by the proposal for a trust-based data aggregation protocol to support network functionality, which enables coexistence with malicious nodes in the network by using their input to reinforce network decisions.

In Chapter 5, we devise a comprehensive model for node behaviour using a hidden Markov model approach, and show that the Baum Welch algorithm can be used to estimate the transition and emission probabilities of a node. Subsequently, it is shown that the use of a long short-term memory RNN facilitates the early classification and extrapolation of the node's behaviour, such that its attack probability can be estimated to a high degree of accuracy even before sufficient observations are collected.

Acknowledgements

I would like to thank, first and foremost, my supervisor Professor Justin Coon for the opportunity to undertake my thesis within his group. His guidance, support and insight has been invaluable to me at every stage of my research, and I am wholeheartedly grateful for all that I have learned from him.

Thank you to the Bristol Research and Innovation Laboratory of Toshiba Europe for generously funding my DPhil, and additionally to Dr. Michael Baddeley and Dr. Adnan Aijaz for hosting me at their offices and giving me the opportunity to collaborate and exchange ideas.

I would also like to thank my colleagues at the Information and Network Sciences Lab, particularly Dr Mihai-Alin Badiu, who has regularly guided my work, provided valued feedback and insight, and has patiently answered my questions throughout my time here. Additionally, I have also had the pleasure of working in the group alongside Dr. James Farmer, Dr. Sunghwan Cho, Dr. Ferhat Yarkin and Dene Hedges, who have regularly taken time out of their day to have fun and insightful conversations with me, which was a welcome distraction during unusual times such as the Covid-19 pandemic.

Thank you to Grahame Faulkner who has always ensured that the research group is a warm, welcoming and productive environment for everyone who has the opportunity to work here. Thank you to Professors Dominic O'Brien and Steve Sheard, for taking the time to read my work during my Transfer of Status and Confirmation of Status assessments, and for providing constructive feedback and insight.

Beyond academic matters, I feel very grateful to have been supported by my friends at Oriel College, at the Oxford University Racing Team, and the Women

in Engineering Society.

I would like to wholeheartedly say thanks to my family, without whom I would not have had the opportunity to complete a doctorate. To my mother, Soma, who has continually been an inspiration to me by pushing boundaries for women in STEM. To my father, Rajeev, who has encouraged and facilitated my education in every way possible. To Didi, whose support has given me the confidence to take on new challenges.

Lastly, I am eternally grateful to my partner, Jonah, whose unwavering positivity and belief in me has underpinned every stage of this thesis.

List of Figures

List of Tables

List of Manuscripts

[Published] T. Deshprabhu, J. P. Coon and M. -A. Badiu, "A Distributed Ledger with Trust-Based Data Aggregation for a Lightweight IoT Network," 2021 17th International Symposium on Wireless Communication Systems (ISWCS), Berlin, Germany, pp. 1-6, 2021.

[To be submitted] T. Deshprabhu, J. P. Coon, "Context-Based Detection of Distributed Network Trust Attacks using a Support Vector Machine".

List of Acronyms

Table 1: Table of Acronyms in Alphabetical Order

Acronym	**Meaning**
ARIMA	Autoregressive integrated moving average
ARQ	Automatic repeat request
AWGN	Average white Gaussian noise
BC	Blockchain
BER	Bit error rate
BW	Baum Welch
CH	Cluster head
DL(T)	Distributed ledger (technology)
DoS	Denial of service
FEC	Forward Error Correction
HMM	Hidden Markov model
IoT	Internet of things
LDPC	Low-density parity check
LSTM	Long short term memory
ML	Machine learning
PoS	Proof of stake
PoW	Proof of work
RNN	Recurrent neural network
SNR	Signal-to-noise ratio
SVM	Support vector machine
TA	Trust attack
UAV	Unmanned aerial vehicle
WSN	Wireless sensor network

Table of Contents

1 | Introduction

1.1 Background

Over the past fifty years, the availability and nature of electronic communication has changed significantly and rapidly. At one time, being able to make a phone call was considered a luxury. Now, not only are communication devices able to deliver information across the world within seconds, but any delay or difficulty in doing so would be considered an anomaly. By 1999, the average household in the UK would have expected to have access to a reliable landline telephone. By 2009 and then 2019, an average UK resident had access to a mobile phone and then a smartphone respectively. In 2024, the landscape of connected communication devices is much more diverse than ever before, having expanded beyond the realm of telephone technology, such that the average UK household has 2.4 people [1] and 9 internet-enabled devices [2].

Aside from phones, common wireless home network devices include smart lighting, televisions, smart home concierge devices, security systems, smart wearables such as VR gadgets and smartwatches, automated environmental control tools such as energy smart meters, and appliances. Outside the home, the majority of industries have come to depend on connected devices in one way or another. For example, wireless sensor networks (WSNs) play integral roles within projects across e-agriculture, shipping, defence, wildlife conservation, environmental studies and many others [3].

This increase in availability of communication devices has led to a vision of future technology that is centred on the concept of distributed networks. When one imagines futuristic communications, it brings to mind networks of seamlessly in-

terconnected wireless communication devices that can exchange information in real-time, autonomously, efficiently, and perhaps most importantly, *securely*. The desire for this level of service can be seen throughout the literature [4] and across a variety of applications, particularly with the advent of the Internet of Things (IoT), such as the vehicular IoT network seen in Fig. 1.1. The idea behind IoT concepts, such as this one, is that the use of peer-to-peer, direct data transfer, in combination with message relaying, would allow for large-scale, real-time communication between nodes in a network. This would allow the vehicles and infrastructural sensors to collectively carry out vital traffic operations, such as traffic flow management, environmental data collection, and many other possibilities [5].

This is an example of a distributed communication network, which allows network nodes to communicate directly with each other. This is in contrast with the traditional communication infrastructure that typically requires network data to travel via a central base station, or hub. Centralised and distributed networks are two ends of a spectrum of network architecture centralisation, as illustrated in Fig. 1.2.

1.2 Distributed Communication Networks: Motivation, Applications and Challenges

There is a growing interest in the design and implementation of decentralised and distributed communication networks. The key factors driving this shift towards decentralisation are:

1. **Network traffic limitations of centralised network infrastructure:** The central node typically has a limited rate of processing requests, meaning that a very large network can often result in a backlog, or network bottle-

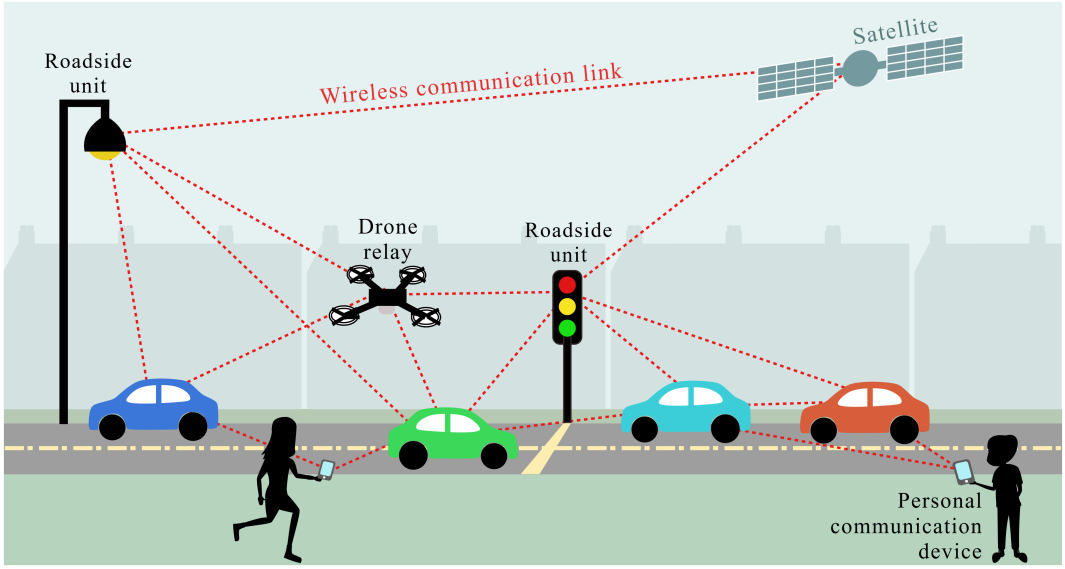


Figure 1.1: Example of a vehicular IoT network concept

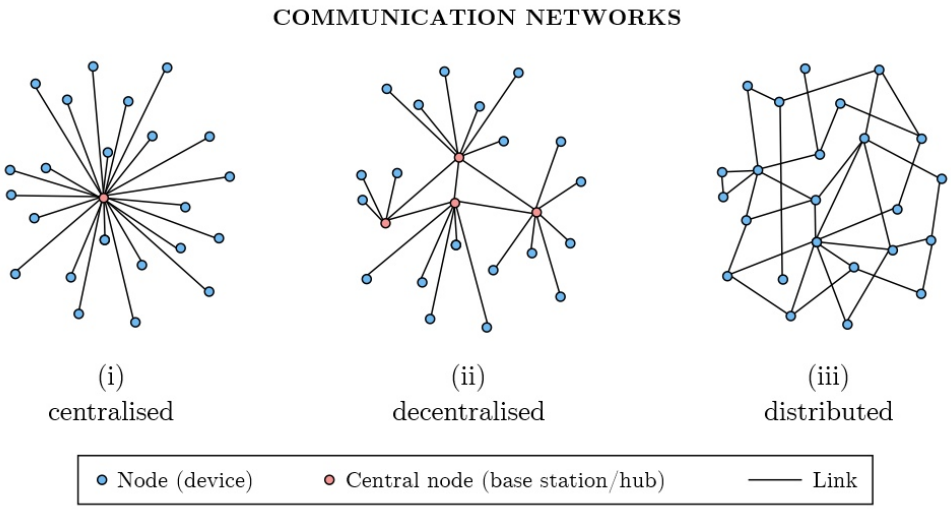


Figure 1.2: Communication network architectures.

neck, at the hub. [6] discusses the estimation of a network's upper limit of service, and the relationship between the rate of requests and the length of the network queue. Secondly, as the network grows, the finite shared channel is spread between more nodes, and the management overhead on the

hub increases. The central hub typically utilises a multiple access protocol to allocate network resources in the most optimal way to satisfy multiple requests. However, a network with more nodes is likely to experience more transmission delays on average than the same network with fewer nodes [7].

2. **Security vulnerabilities of centralised network:** There is a significant dependence on the hub in a centralised network for vital network functions. As a result, the hub is a major vulnerability of the network, and any failures in its operation would disrupt the functionality of the whole network [8]. This is perhaps the most challenging aspect of centralised networks, since this vulnerability can be taken advantage of by malicious parties, as is discussed further in Chapter 2.

3. **Need for scalability, beyond what is available in centralised networks:** With an estimated 50 billion IoT devices in the world by 2025 [9], scalability of networks has become a priority. In theory, distributed networks have the ability to scale not only in terms of the number of connected devices, but also geographically. This is because, through the relaying and diffusion of data across nodes, distributed networks can allow for communication over large distances. This is particularly useful for collecting and transferring data in remote environments, where it is not possible for all nodes to be within reach of a base station [8].

4. **Consumer desire for automated, real-time systems:** The consumer view of future telecommunications is centred on the idea of seamless and automated communication services. [10] goes into detail about the consumer needs, trends and wants, as well as the industry-scale desire for reliable automation. Since the implementation of 5G cellular networks, and even more so with the expected terrabits-per-second (Tbps) speeds of 6G net-

works [10], the speed and reliability of the technology is now capable of supporting futuristic ideas of IoT networks in the home, as well as across industries. However, the infrastructure needs to be updated accordingly to support these anticipated applications, and it is expected that distributed networks are the solution.

The successful implementation of distributed communication networks would enable a vast range of opportunities, particularly in the development of WSNs. WSNs comprise many wirelessly interconnected sensor devices that collectively gather and process data about their environment. This is more scalable than the centralised equivalent, for which all nodes would need to be within communicable distance of some common point, i.e., the central hub. Other anticipated distributed network applications include as for real-time monitoring of conditions across large expanses of agricultural land [11], vehicle-to-vehicle networks [12], drone response in disaster management [13], instant cross-referencing of medical records between healthcare providers [14], and countless others.

All of these applications will include the exchange of sensitive information between nodes. In the highlighted example applications, this may include location data, sensor readings, financial data, classified messages, medical records, etc.

1.2.1 Network Attacks

Data is valuable. It can and has been exchanged as a form of currency in itself, and is particularly appealing to those who may take advantage of the information found within network data. As a result, communication networks are commonly targeted by malicious parties for the purpose of accessing valuable data or disrupting network functionality. [15] reports over 4500 data breaches and network attacks globally in January 2024 alone. Owing to the lack of an overseeing au-

thority, a distributed network's security needs to be somewhat autonomous. There is a need for distributed protocols to manage vital network tasks such as data validation, version control, and node authentication, the network could potentially become vulnerable to attacks.

John von Neumann first theorised the idea of the computer virus in the 1940s [16], describing it as an automatically self-replicating entity about thirty years prior to the creation of the first ARPANET virus, the Creeper Worm [17], in 1971. Following this, the 1888 Morris Worm led to severe disruption of over 10% of all internet-connected devices within the first 24 hours of its release [17], and prompted the development of the first coordinated cybersecurity effort via the establishement of the CERT Coordination Centre at Carnegie Mellon University.

Since then, both the network security and network attack protocols have continuously fought to stay one step ahead of the other. Network attacks have diversified beyond self-replicating software, and successful attack protocols are now highly tailored to their specific purpose and environment. In addition to attack complexity, the average impact of an attack on a network has continually increased. [18] estimates that the mean financial cost of recovering from a single network attack or data breach is four and a half million US dollars.

The need to continually stay one step ahead of malicious attacks has resulted in an established field of research in network security, post-attack recovery, and device protection. In order to mitigate adverse financial and functional effects of network attacks, it is preferable to take a defensive and preventative approach, instead of focusing on recovery alone.

1.2.2 Research Trends in Distributed Network Security

With the research and technology moving towards distributed networks, there are concerns that this new network infrastructure will facilitate a variety of new network attack protocols. Some distributed network attack protocols have already been theorised or extrapolated from existing communication systems, which is expanded on in Chapter 2. However, in addition to targeted attack protocols, the literature has identified the following vulnerabilities in distributed networks:

1. It takes time to deliver or flood data across the network, meaning that some nodes may have outdated versions of network data. This is particularly true for large distributed networks [19].

2. Decision-making cannot be done by a single entity in a distributed network, and therefore collective decision-making needs to be integrated [8].

3. Centralised security protocols cannot be directly applied to distributed networks due to the lack of central authority in the latter, resulting in a need for restructured security protocols [19].

4. Distributed networks are particularly useful in applications where there may not be sufficient infrastructure to support a centralised alternative. However, this can also mean that the network links are intermittent, for example where the nodes are moving or in adverse weather [20].

Many of these risks can be attenuated by ensuring that nodes across the network have, and can maintain, a unified view of the data. This requires the careful design of protocols for distributed network flooding, version control, and distributed decision-making protocols called *consensus* [21]. Table 1.1 provides the highlights from this research by comparing the four most prominent consensus algorithms within distributed network research.

Table 1.1: Summary of relevant consensus algorithms.

Consensus Algorithm	Summary	Advantages	Disadvantages
Proof of Work (PoW) [22]	Nodes race to complete an intentionally difficult cryptographic puzzle.	Low incentive for dishonest mining because this would waste energy.	Requires more time and energy than available for most IoT nodes, high transaction latency.
Proof of Stake (PoS) [22]	Chooses the successful miner (data validator) by determining which miner has the most stake in the network, e.g. which miner has been participating for the longest time.	Makes mining more accessible to less capable nodes, malicious nodes must have stake before they can have an effect on the network.	Malicious nodes may also have a stake in the network, which would make them indifferentiable from other nodes.
Paxos [23]	First fault-tolerant consensus algorithm. Its overall function is to take a majority vote.	Extremely fault tolerant, including for Byzantine faults.	Complex algorithm leads to high transaction latency, not suitable for nodes with limited processing power.
Raft [24]	Simpler form of the Paxos voting system which uses a leader-follower structure.	Designed to be a less computationally intensive version of Paxos, achieves comparable performance and fault tolerance.	Leader-based structure is centralised, poor performance with Byzantine faults.

However, consensus is only one part of ensuring secure distributed communications. With the use of consensus protocols, some papers explore the allocation of network responsibilities to different nodes dynamically. This takes advantage of the flexible nature of distributed communications by ensuring that important responsibilities are always placed upon nodes that have, for example, a reliable connection to the network or are closest to the destination at that time.

For example, [25] surveys task assignment methods for unmanned aerial vehicles (UAVs). It highlights a method for optimising resource allocation based on the abilities, location and resources of each node in the UAV network. This is a suitable approach for UAV networks, since it would be reasonable to assume that nodes across this kind of network would have comparable abilities, and be

deployed by cooperating parties such as aid organisations in a natural disaster. However, when the nodes are not necessarily familiar to each other or where there is a possibility that certain nodes have been compromised, the task assignment approach presents the risk of assigning sensitive functionality or data storage to a malicious node.

Another approach has been to rely on cryptographic validation for providing security to the network transactions, such as distributed ledger technology (DLT), which relies on *zero-trust protocols* for distributed security. These protocols enforce validation and verification of network participants, as well as the network data. The principle behind zero-trust is that no network entity is to be trusted without thorough verification, and in order to achieve this, the protocols often employ highly demanding computational tasks of the nodes being verified. Zero-trust protocols, such as Proof-of-Work (PoW) [28], are currently utilised widely across existing distributed communication networks, but mainly within cryptocurrencies such as the BitCoin blockchain network [29]. Within financial networks, zero-trust protocols are used to facilitate distributed consensus.

For a time, there was an especially pronounced interest in the use of blockchain technology across the technical sector [26], including distributed communication networks, but this was quelled by the highly demanding nature of the blockchain protocols. This is evidenced by the fact that Proof-of-Work (PoW) [28], as discussed in Table 1.1, cannot currently be executed by typical consumer communication devices, e.g. laptops, mobile phones, etc. This is why the research has broadly come to an agreement that the blockchain protocol cannot directly be applied to distributed communication networks, though there is value in the accountability afforded to a public network by the inclusion of a permanent ledger. Some have addressed the overhead issue by delegating different levels of service to different nodes using a hierarchical approach. For example, [27] only provides

full functionality to nodes that have proven their capabilities.

We can see from these research trends that there is a common theme in the literature of wanting to ensure that vital network functionality is not entrusted to untrustworthy parties. For example, resource allocation would typically favour trustworthy nodes, whilst message relaying would favour routes that avoid untrustworthy nodes.

This leads us to the question of: how can the network nodes ascertain which of their peers actually is *trustworthy*?

1.2.3 Trust As a Security Metric

With the advent of IoT and WSNs, it has become clear that there needs to a be distributed security protocol that can achieve consensus but without the highly computationally demanding requirements. This is because the nodes involved in IoT systems are expected to be restricted by limited computational capabilities and, depending on the application, limited energy, unreliable quality of communication link, and highly intermittent internode connections. [30] explores the current expectations and limitations of future IoT networks and their nodes.

Trust inference has emerged as a possible solution to distributed network security where zero-trust protocols are not suitable, particularly in networks with limited resources, a preference for low overhead, or requirement for low latency [31]. Trust alone is not a security metric, but instead allows network agents to evaluate each other based on observations. The inference algorithm is responsible for mapping from the observation set to a trust metric, which can then be used as a foundation upon which security protocols can be designed.

Early examples of trust-based networks can already be seen in operation. For example, user ratings were commonplace in internet forum communities by the

early 1990s, and was formalised as a user reputation system by eBay in its online marketplace in 1995 [32]. Research interest in trust inference for communication networks has steadily grown since around 2002, when [33] presented a pivotal perspective on Bayesian inference of trust and reputation based on observations, by modelling trust probability as a beta-distributed random variable.

Trustworthiness, or simply *trust*, metrics provide a valuable tool in distributed networks for the following reasons:

1. Trust inference algorithms allow network nodes, regardless of their resources, to independently evaluate each other and the network data based on observational data.

2. Managing trust data is a significantly smaller overhead on the network when compared to the cryptographic solutions. Trust data can be stored locally at each node, or the network can consider facilitating a Trust-over-IP approach, which integrates an additional network stack layer specifically for the management of observational and trust data [34].

3. Observations can be exchanged or relayed across the network in order to quickly build up evidence about all nodes across the network. [33] discusses reputation discounting, which statistically accounts for the validity of relayed observations based on the reputation of the relaying node from the receiver's perspective.

4. The available inference protocols are highly adaptable for specific system requirements. For example, where recent node behaviour should bear greater weight in its reputation, this can be easily integrated into the protocol via a *forgetting factor*, as used in [33].

Trust inference provides a promising solution to distributed network security, and

the design of inference protocols is already an established and vastly growing field of research, with many inference algorithms available in the literature. Therefore, this thesis does not attempt to redesign the inference protocols, but instead, we focus on the remaining challenges in the implementation of trust inference:

1. As communication systems do not currently employ trust inference in their implementation, there is limited-to-no real-world data available about trust inference networks. Instead, most of the existing literature relies on generating realistic training data [35].

2. A need to utilise trust information in the design of distributed network self-organisational tools. This would support secure distributed data exchange, clustering, resource allocation, data validation, and other core network functions in the absence of a central authority.

3. To better understand the nature and scope of trust network vulnerabilities, such as insider attacks. The existing literature has explored a wide variety of possible trust attacks (TA), and has proposed a number of targeted security protocols. However, the view of TAs is disjointed due to the lack of any standardised framework for evaluating threats in a trust network.

4. There is a need for further research into how a network can preempt TAs in order to intercept or minimise their adverse effects on the network, rather than simply controlling damage after-the-fact.

1.3 Research Project

1.3.1 Summary of Research Problem

The general trend of communication networks is towards utilising distributed architectures in order to alleviate the dependence on a central network authority, and allow for scalability. However, new network architectures come with new security challenges. As discussed in Section 1.2.3, trust inference is a strong premise for distributed network security since it meets the majority of needs by allowing network nodes to both independently and collectively evaluate each other's reliability, without highly demanding computation.

However, within trust inference itself, there are a number of vulnerabilities that need to be addressed, such as insider attacks. Furthermore, instead of protecting the network against known attacks by designing individually targeted defences, it would be beneficial to design mechanisms by which malicious activity can more comprehensively be predicted and prepared for. If it is known in advance that a network node will attack at a given time, and using a given protocol, then its effect could be minimised through mitigating measures by the rest of the network.

1.3.2 Aims & Research Questions

The aim of the thesis is to understand and address security vulnerabilities within trust networks, specifically with a view to coexistence with malicious nodes and the prevention of trust network attacks in distributed communication systems.

This question can be divided into a set of core research questions, where each of the questions 1-3 correspond to our 3 research chapters respectively:

1. **Framework of threats:** What are the threats within a distributed, trust-

based network?

(a) What are the foremost security risks to be aware of in trust networks, and how can they be effectively detected and classified?

(b) From a security protocol perspective, what are the symptoms of trust network attacks, and how can those symptoms be mapped to impending network attacks preemptively such that the attacks' effects can be minimised?

(c) Can the network utilise knowledge about the nodes and the wider range of possible threats in order to protect itself, without needing individual tailored defense mechanisms for each possible attack?

2. **Coexistence with threats:** Can trust inference data enable the secure self-management of distributed network functions, in preparation for proposed future applications like vehicle-to-vehicle communication?

   (a) Is it plausible for a trust network to coexist with known malicious nodes by leveraging the knowledge of malicious behaviour in order to reinforce network decisions, functions and data?

   (b) Equipped with trust data, how can nodes in a distributed communication network exchange and collectively carry out vital network functionality in a secure and distributed way, specifically, secure data storage and exchange?

   (c) How can trust inference be integrated into the design of distributed network protocols?

3. **Prediction of threats:** Is it possible to restructure and process a network's trust data in order to gain insight about potential future TAs?

(a) How can a network node be modelled in a way that facilitates the temporal extrapolation of node behaviour, in order to identify malicious nodes as early as possible?

(b) Can that model be designed in such a way that it is comprehensive for most, or all, trust-based attacks?

(c) Is it possible to estimate the trust of a node based on early interactions within the network?

(d) Can those early interactions be analysed to predict next-step or near-future behaviour of a node?

1.3.3 Considerations

Given that trust inference has come to prominence in response to the need for a distributed security framework for devices that would not be able to utilise the alternative, more computationally intensive, cryptographic protocols, it follows that the project should consider the same end of the device spectrum. In other words, we broadly take into account the suitability of any proposed network protocols for application within IoT and WSN networks.

One of the major challenges across the field of trust research is that there is very limited data from existing trust networks, since these are not yet widely implemented. Therefore, where training data is required, it is common practice to generate realistic training data based on probability distributions. This is done independently of any processing of that data in order to maintain realism and integrity of subsequent simulations.

Note that the specifications of the system used to complete this thesis have been provided in Appendix A.

1.3.4 Thesis Overview

Chapter 2 provides background theory about trust inference, as well as the main tools and concepts utilised in the research.

In Chapter 3, we investigate Research Question 1 posed in Section 1.3.2 by designing a unified framework of trust network attacks. This is then used to propose a novel contextual characterisation method for the classification of malicious network nodes into possible attack scenarios. Finally, the contextual characterisation is used to classify nodes using a Support Vector Machine (SVM) learning algorithm.

Research Question 2 is examined in Chapter 4 via the design and analysis of a trust-based, distributed data management protocol for wireless communication networks. This is accompanied by a proposal for a novel trust-based data aggregation protocol to support network functionality, which enables coexistence with malicious nodes in the network by using their input to reinforce network decisions.

In Chapter 5, we devise a comprehensive model for node behaviour using a hidden Markov model approach (HMM), and show that the BW algorithm can be used to estimate the transition and emission probabilities of a node. Subsequently, the change of the BW estimate over time is used as temporal data to predict the attack transition probability of a node during its initial interactions in the network, with the aim of classifying potentially malicious nodes as early as possible. This early classification is implemented using a Long Short Term Memory (LSTM) recurrent neural network (RNN), which is then adapted for prediction of the node's next-step and near-future attack transition probabilities. This research addresses Research Question 3 posed in Section 1.3.2.

2 | Literature Review

This chapter covers the background theory and relevant literature behind the work presented in Chapters 3-5. We begin with a brief summary of packet transmission and error checking, followed by an overview of trust theory, and finally a description of all the theoretical tools used as part of our research. Note that any theory or principles that we have manipulated or built upon are presented directly within their respective chapters.

2.1 Context: Transmission and Error Checking

In this section, we give a brief and general overview of a typical network set-up that would benefit from the research being conducted in this thesis.

2.1.1 Packet Transmission

Packet switching is the transmission of data in the form of packets over a digital channel. Data packets contain header bits, followed by data bits, ending with trailer bits. The expected format and length of the packet's contents are determined by the protocol being used.

The alternative, circuit switching requires a direct physical layer connection between the source and the destination, resulting in data arriving in the correct order, consistent data rate, and guaranteed bandwidth. In contrast, packet switching makes more efficient use of the channel capacity by allocating a route for each packet on an ad-hoc basis. The routing protocol can be optimised for resource efficiency, latency, reliability, and a wide variety of other parameters. Importantly, packet switching protocols incorporate fault tolerance by allowing for the packet to be re-sent if lost or damaged. Damage can occur as the result of a poor channel or as the consequence of a malicious network attack.

Fig. 2.1 shows the block diagram for a typical wireless transmitter and receiver. Network attacks have the potential to occur at multiple points in this process, particularly at the source, if the source itself is malicious, or within the channel, if the communication channel is not secure. Note that the channel model being used in our work includes additive white Gaussian noise (AWGN), and this, along with any further disruption to the channel performance, is represented by a constant channel error probability in numerical simulations.

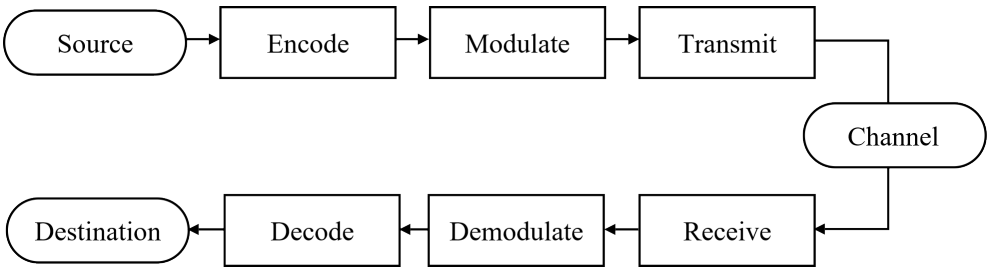


Figure 2.1: Wireless network transmitter and receiver.

Regardless of the reason for a damaged packet, error correction can be incorporated into the packet's format in order to (i) determine that damage has occurred, and (ii) attempt to reverse that data.

When the error involves the arrival of packets out of chronological order, the receiver can request a re-transmission of the next expected packet using a range of protocols that are referred to as automatic repeat request (ARQ) [36]. However, ARQ only facilitates the re-transmission of the packet, rather than the recovery of partially damaged packets. Therefore, in the latter case, ARQ would be inefficient due to reusing the channel resources even if only one bit in the packet is damaged.

Secondly, forward error correction (FEC) [37] involves adding redundant *parity* bits to the packet, which can be used at the receiver to both determine which bits are potentially damaged and also attempt to reverse engineer those bits. Some of the widely used FEC protocols include convolutional codes, which incorporate the parity bits amongst the message itself and require memory, and block codes, which add a sequence of parity bits to the end of the message, and are memoryless. Typically, block codes are preferred for data that arrives in bursts, such as WSN data, whereas convolutional codes are preferred for long streams of data.

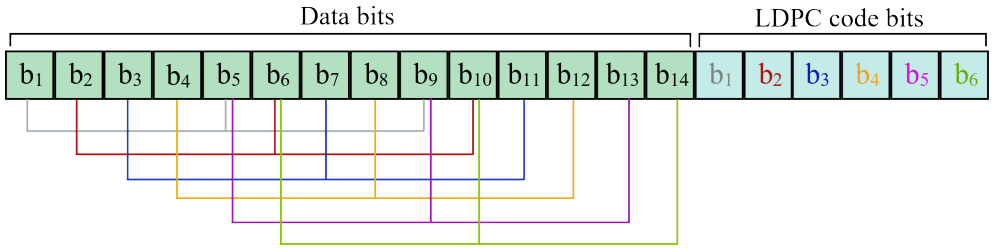


Figure 2.2: Illustration of LDPC parity check and code generation.

2.1.2 Low-Density Parity Check Codes

There are many different types of block codes available. We have chosen to use low-density parity check (LDPC) code [38]. This is because LDPC code exhibits a lower BER than common alternatives such as Hamming and Reed Solomon codes. Secondly, although the BER of LDPC is marginally higher than that of polar codes, LDPC decoders are much less complex and therefore result in higher throughput and lower latency [39].

The encode and decode operations ensures the privacy of the data, and the protocol for these are known to both the source and destination. A common encoding protocol is the LDPC code which systematically groups bits from the data, performs a parity check on each group, and appends the resulting parity bits to the end of the packet. An example of this process is illustrated in 2.2.

The coded data is then modulated and transmitted across the communication link. At the receiver end, once the received packet has been demodulated, the LDPC decoder can use the LDPC code bits to identify which of the received bits has been flipped between the encoder and itself. Packet transmission and LDPC error checking are used in Chapter 4.

The improvement of BER due to the use of LDPC code is shown in the simulation results in Fig. 2.3.

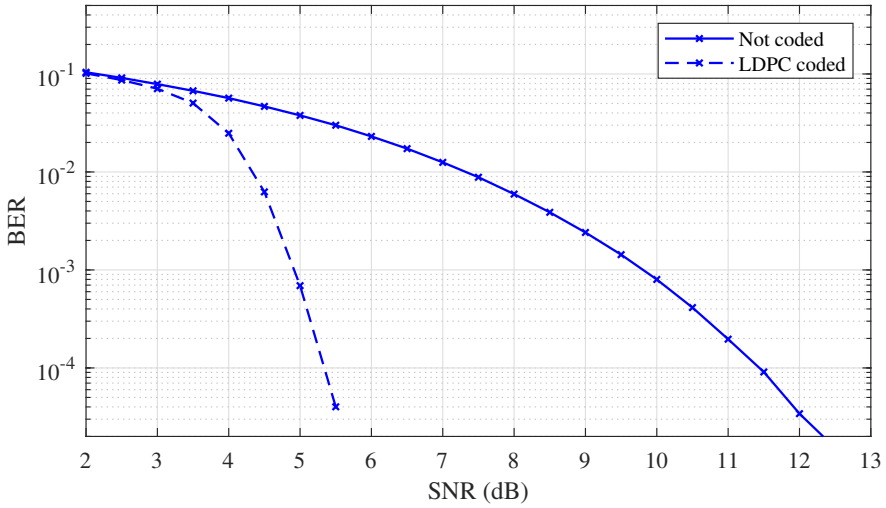


Figure 2.3: Simulation BER results for LDPC code vs. uncoded data transmission. The data is QPSK modulated and transmitted across an AWGN channel.

2.2 Trust and Reputation

The field of trust inference research deals with numerical calculations and representations of an agent's predictability. Naturally, this predictably translates within the communication network environment as a measure of reliability, or trustworthiness, of network nodes.

Let the definition of trust be as follows:

Definition 1. *The trustworthiness, or trust, of a target is the probability that, the next time the observer interacts with that target, it will behave as the observer expects it to.*

The process of integrating trust metrics into a system requires consideration of three key areas:

- **Evidence:** The collection of observations about an agent. What is defined as a positive or a negative observation depends on the particular protocol.

However, in line with the idea of predictability, positive observations are typically considered to result from interactions that resulted in the *expected* outcome.

- **Inference:** A scheme for mapping observations from evidence space to the trust space by inferring a trust probability $\tau \in [0, 1]$ from a set of positive r and negative w interactions with the target node. This stage may additionally involve the conversion of trust from a probability into a global representation called reputation. A simplistic reputation model would be to scale the trust probability τ such that the reputation is centred on 0, i.e., the reputation metric $\varphi \in [-1, 1]$.
- **Response:** Define how the network protocol should respond to the inferred trust. Specifically, responses aim to prevent untrustworthy nodes from having a significant, adverse effect on network functions. The most common response model is to either exclude or attenuate input from nodes with low τ [40], sometimes using a threshold-based approach for determining the weight of attenuation. Another common approach in literature is for only the low trust nodes to be monitored by the network, such that the network need not monitor activities from all nodes in the network [40].

2.2.1 Trust Theory

A widely regarded benchmark for the development of new trust inference models is the Beta Reputation System [33], which models trust p as a beta-distributed random variable. The beta probability density function is

$$f(p \mid \alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} p^{\alpha-1}(1-p)^{\beta-1}, \tag{2.1}$$

where $0 \leq p \leq 1$, $(\alpha, \beta) > 0$, and α and β are the shape parameters of the distribution. The mean probability of p is then

$$E(p) = \frac{\alpha}{\alpha + \beta}. \tag{2.2}$$

The shape parameters broadly translate into observations within the trust domain, and based on this idea, [33] proposed the Beta Reputation System such that node X's reputation function for target node is given by the beta reputation function [33],

$$\varphi\left(\tau \mid r_T^X, w_T^X\right) = \frac{\Gamma(r_T^X + w_T^X + 2)}{\Gamma(r_T^X + 1)\Gamma(w_T^X + 1)} \tau^{r_T^X} (1 - \tau)^{w_T^X}, \tag{2.3}$$

where r_T^X and w_T^X correspond to total positive and negative observations respectively by X about T, and τ represents the trust probability of T. Note that $r_T^X + 1$ and $w_T^X + 1$ therefore represent the future totals of positive and negative observations, as it is the *next* interaction being evaluated.

Following (2.2), the expected trust probability becomes

$$E(\tau) = \frac{r_T^X + 1}{r_T^X + w_T^X + 2}. \tag{2.4}$$

Note that when $r = 0, w = 0$, the initial expected trust probability is $E_0(\tau) = 0.5$, and therefore $E(\tau) < 0.5$ and $E(\tau) > 0.5$ would not necessarily be considered the benchmarks for low and high expected trust probability respectively. This is because, in practice, it would be reasonable to set the condition for high trust to $E(\tau) > 0.9$. This is reflected in existing rating systems, where the average eBay seller has a reputation of 0.98 [32].

2.2.2 Trust and Insider Attacks

The majority of trust response protocols are designed to attenuate or exclude input from untrustworthy nodes, or to limit the scope of activity for untrustworthy nodes [40]. A combination of these has been used in [41], where the level of service obtained by a node in the network increases as its reputation improves.

As a result of these responses, it would be logical for a malicious node to first cooperate with the network in an attempt to increase its perceived trustworthiness, prior to committing the attack. This occurs regularly in online marketplaces, where malicious users may obtain artificially inflated reputations via collusion with other dishonest users [42].

It is expected that similar strategies would be utilised by malicious parties in distributed communication networks. A more detailed survey of TAs has been conducted as part of the framework in Chapter 3, and we have highlighted some common TA strategies below:

- **Maintenance of high trust:** A node with high trust or reputation will typically have a greater effect on the network than a node with low trust. This is the basis of the ON-OFF Attack [43].
- **Maintenance of neutrality:** A node with neutral trust and one with relatively few past interactions in the network are indifferentiable from each other based on the observation set alone. Therefore, it becomes difficult to penalise untrustworthy nodes if they continually leave and re-join the network in order to reset their perceived trust. This is the basis of the Whitewashing Attack [44].
- **Power in numbers**: A network's access to accurate trust data about its nodes can be adversely affected by the presence of many malicious nodes,

particularly if they are colluding with each other. The trust inference of any given node can be obscured if there is an abundance of false positive observations being reported and relayed by colluding nodes across the network. Collusion attacks in trust networks are explored in [45]. This strategy is evident in the *51% attack* on cryptocurrency networks, whereby if over half of network nodes are compromised by a malicious party, then that party has the majority decision-making power in the network [46].

2.3 Machine Learning, Recurrent Neural Network, and Estimation Tools

This section gives an overview of the background knowledge and theory behind any statistical analysis tools used in the subsequent research chapters. Any algorithms that have been customised or adapted in any way are detailed within the research chapters directly.

2.3.1 Support Vector Machine and K-Means Clustering

Machine learning (ML) algorithms can be divided into two categories, supervised and unsupervised. Supervised ML algorithms, such as SVMs, are trained using *training data*, which consists of labelled data points. Then, the resulting trained learner can be applied to fresh *test data* to obtain the most likely labels for each new datapoint. This process is used to classify data, or map from datasets to labels, such as image recognition. Unsupervised ML algorithms, such as K-Means Clustering, are not trained using labelled training data, and are typically used for analysis of datasets such as clustering, sequence detection, and reducing the dimensionality of the data.

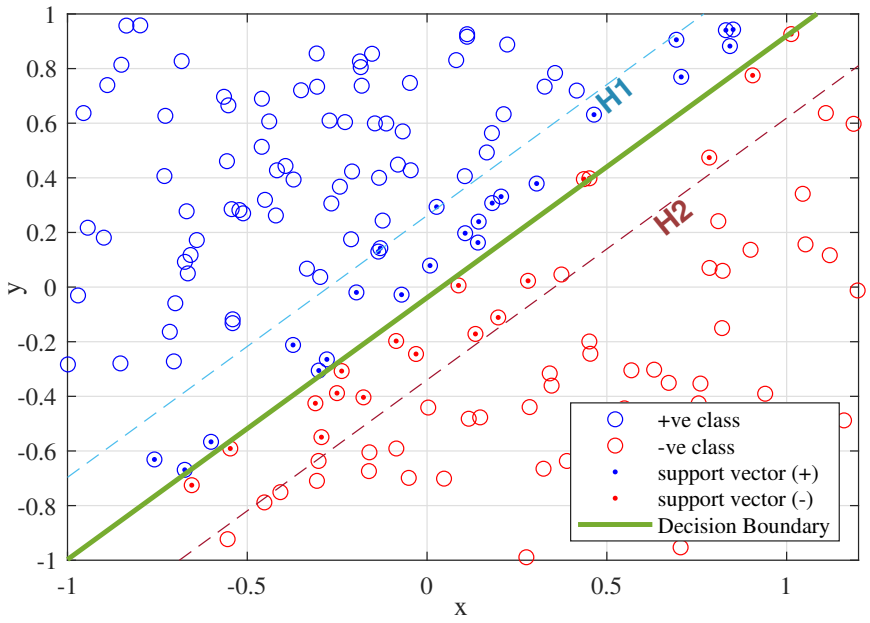


Figure 2.4: Demonstration and illustration of SVM classification on Matlab's *'Linearly Separable Data'* sample dataset

We utilise both the SVM and K-Means Clustering algorithms in our work in order to classify and cluster data about network nodes.

With respect to the linear example in Fig. 2.4, which demonstrates a simple SVM classification using Matlab sample data [47], the classifier seeks to find the Decision Boundary such that its distance from the margins $H1$ and $H2$ is maximised. A hard margin is where there are no datapoints of its respective class between itself and the decision boundary. However, the example in Fig. 2.4 includes soft margins due to the existence of support vectors between $H1, H2$ and the decision boundary.

Considering the linear equation $y = a \cdot x + b$, which can be rearranged as $a \cdot x + b - y = 0$, the decision boundary can be defined in vector form as

$$\boldsymbol{W} \cdot \boldsymbol{X} + b = 0, \tag{2.5}$$

where $\boldsymbol{X} = (x, y)$ and $\boldsymbol{W} = (a, -1)$. The distances from the margins $H1$ and $H2$ to the origin are, respectively,

$$H1 : \boldsymbol{w} \cdot \boldsymbol{x} + b = +1, \tag{2.6}$$

$$H2 : \boldsymbol{w} \cdot \boldsymbol{x} + b = -1. \tag{2.7}$$

The margin to be maximised is the difference between these distances,

$$M = \frac{2}{|\boldsymbol{w}|} \tag{2.8}$$

Updating the variables can be treated as an optimisation problem using Lagrangian multipliers l,

$$l = \min \frac{\|\boldsymbol{w}\|^2}{2} : y_i = (\boldsymbol{w} \cdot x_i + b) - 1 \geq 0 \tag{2.9}$$

$$= \frac{\|\boldsymbol{w}\|^2}{2} - \sum_{i=1}^{l} \lambda_i (y_i(\boldsymbol{w} \cdot x_i + b) - 1) \tag{2.10}$$

By solving for $\boldsymbol{w}$, λ and b, we can define the bound of l as:

$$\max l_d = \sum_{i=1}^{l} \lambda_i - \frac{\lambda^T \lambda K}{2}, \tag{2.11}$$

where λ^T represents the transform of λ, and K represents the kernel function used to map (x_i, y_i) into a higher dimension such that the data becomes separable. This process can be programmed based on the following pseudocode:

Where there is no clear separating hyperplane for data, the SVM transforms the dataset into a higher dimension such that it becomes more clearly separable. This can be done using a *kernel function*.

Algorithm 1 Find w and b

if $y_i = (w \cdot x_i + b) - 1 = 0$ **then**
  (x_i, y_i) are the support vectors,
  keep w, b values,
else if $y_i = (w \cdot x_i + b) - 1 > 0$ **then**
  keep w, b values,
else if $y_i = (w \cdot x_i + b) - 1 < 0$ **then**
  update w, b values.
end if=0

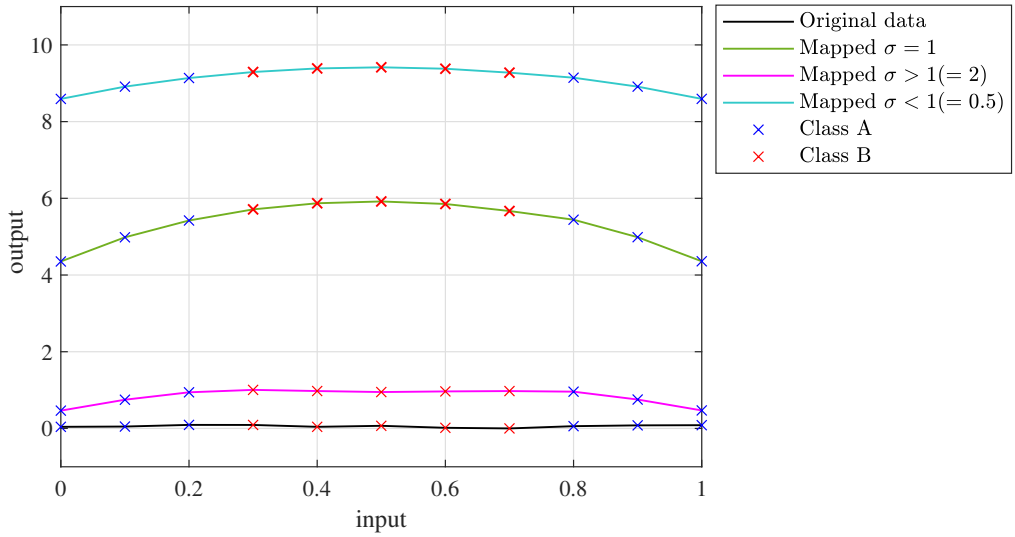


Figure 2.5: Gaussian kernel mapping applied to linearly inseparable data with $\sigma = 1$ (unscaled), $\sigma = 0.5$, and $\sigma = 2$.

Fig. 2.5 provides an illustration of how a Gaussian kernel function can transform linearly inseparable data such that it becomes linearly separable.

Similarly, the related unsupervised ML algorithm, K-means clustering, aims to identify decision boundaries between clusters of data. In contrast with SVM, the K-means approach takes a *test-and-update* approach rather than being trained using labelled data. It chooses K random data points as initial cluster centroids, and then assigns the remaining datapoints to their respective nearest centroids,

$$\arg\min_{c_i \in C} \textit{dist}\left(c_i, p\right)^2, \tag{2.12}$$

where c_i represents the chosen cluster centroid location and p represents the data point. Once all nodes are categorised, the mean squared distance is calculated between all data points and their assigned centroids. The centroid locations are updated by finding the centre of all the data points **x** in each category S_i,

$$c_i = \frac{1}{S_i} \sum_{x_i \in S_i} x_i. \tag{2.13}$$

This process is repeated over a specified number of iterations, with the system finally choosing the centroids that produced the lowest overall square mean distance.

2.3.2 Baum Welch Algorithm for Hidden Markov Models

The BW algorithm estimates the transition and emission probabilities for a HMM, $\theta = (C, D, \pi)$, where C and D are the state transition and emission probabilities respectively, and π is the prior probability distribution of the hidden states. The estimation process consists of an iterative expectation maximisation operation on θ, such that the output is the local maximum,

$$\theta* = \arg\max_{\theta} \mathbb{P}(Y|\theta), \tag{2.14}$$

where $Y = (Y_1 = y_1, Y_2 = y_2, ..., Y_T = y_T)$ represents the observation sequence from time $t = 1$ to $t = T$.

The procedure combines two complementary HMM analysis techniques called the

forward and backward algorithms, which deduce properties of the HMM based on $(Y_1, ..., Y_t)$ and $(Y_t, ..., Y_T)$ respectively, where $t \in [1, T]$. A full description of the algorithm is provided in Appendix B. The BW estimator can be trained using one or more sequences of observations from the HMM. It is expected that the accuracy of the estimate will improve as it is trained using more observation sequences.

2.3.3 Long Short Term Memory

An LSTM is a type of RNN, which is used primarily to analyse, classify and extrapolate time series data. One of its main advantages over a regular RNN is that the LSTM is considerably less prone to the vanishing gradient problem [48]. The vanishing gradient problem occurs for RNNs with many layers as the back-propagation process, used to determine the optimal function weights, results in many multiplications of the derivatives of each layer's activation function. Further background about the LSTM RNN and the underpinning algorithm is given in Appendix C.

2.4 Smoothing Techniques

Machine learning and estimation tools are rooted in the statistical analysis of data's trends and unique properties. Prior to applying these tools, it is often beneficial to pre-process the data. Smoothing, in particular, is well-suited for this purpose because it helps to retain the data's trends and other key properties, whilst attenuating any random noise, jitter, or other unwanted artefacts in the data.

Two commonly used smoothing techniques are moving average (μ_{MA}) smoothing and exponential smoothing (s), the equations for which are given in Appendix D.

3 | A Support Vector Machine Approach to Early Characterisation of Trust Attacks in a Distributed Network

Abstract

During the literature review stage of this project, it became apparent that there is a significantly unclear view of the range of trust-based network attacks. This is largely due to the existence of minor variations of similar attack protocols, disjointed terminology from paper to paper, and a lack of any kind of systematic threat framework.

In order to design suitable security protocols based on trust, it is essential to first design a unified framework of attacks. Instead of simply listing examples from the literature, we have collated a large number of attack protocols into an intuitive, logical and security-focused framework that is designed specifically from the perspective of an observer. Based on this, we propose a contextual characterisation method for the generation of realistic training data about trust networks, as well as for the classification of malicious network nodes based on their strategy.

3.1 Introduction

Trust inference techniques have the potential to facilitate security in distributed communication networks [56]. In contrast with alternative distributed security methods, trust inference protocols are relatively less demanding in terms of their computational complexity, meaning subsequently that they do not incur high transactional latency and are suitable for networks without access to high-capability nodes. Trust inference is already used informally via user rating systems in a variety of applications, such as online retail platforms, for users to independently rate and evaluate each other [57]. However, trust inference protocols still have security vulnerabilities that need to be both understood and addressed.

Trust- or reputation-based security protocols allow users to protect themselves, for example, by only interacting with highly-reputed peers. Having recognised this, malicious parties have been known to invoke ways of increasing their own rating through manipulation of the trust inference system - this is called a *trust attack* [58] (TA).

A TA is a malicious attack within a trust-based network, whereby the attacker seeks to maximise its adverse effect on the network by manipulating the trust metrics prior to the attack. In line with the majority of trust protocols, a higher reputation would allow a node to have a greater effect on the network functionality and group decisions. One solution to this problem is to use artificial intelligence (AI) to enhance malicious node detection and attack prediction. Supervised machine learning (ML) algorithms, such as support vector machines (SVM), can be used to categorise new data based on past and similar scenarios [59, 60]. Applying this to observe and characterise node al patterns may enable malicious activity to be flagged early, such that a preventative response can be implemented, rather

than simply managing the consequences post-attack.

3.2 The Problem

In order to design a trust-based security protocol, it is necessary to have a logical and global view of trust network threats. There is widely varying terminology being used across the literature to refer to TAs, including hundreds of minor variations on the core attack models. This has led to an unclear view of the possible TAs that a network should be able to mitigate. As a result of this, many TA prevention models tend to focus on detecting a specific risk, such as a particular mode of attack. A targeted approach typically exhibits high accuracy in detecting that particular attack but, in practice, it is highly likely that malicious nodes will not only be aware of the security framework by which it is being evaluated but it may also be able to change their strategy to evade detection, as represented by the malicious node model in [61].

Therefore, there is a need for an attack detection protocol that is adaptable to a wider range of attacks. It should be able to consider, for example, if the malicious node is exhibiting symptoms or characteristics of any particular attack strategy and adapt its response accordingly.

Some reviews of TAs from a more general perspective are available in [56,62,63]. However, these are written from an observer's viewpoint, with a focus on the intended outcome of the attack, rather than grouping by procedure. There is a need for a unified TA framework that considers the detector's viewpoint, with a focus on attack symptoms, such that the framework can be used to reinforce malicious node detection. For example, what behavioural signs should the detector be looking for in a node that may indicate a particular attack strategy? Are there various attacks that require the node to manipulate its trust using the same steps, such as

frequently leaving and rejoining the network to refresh its reputation or exchanging falsified positive observations with another node?

Many of the attacks discussed in the literature rely heavily on the malicious node already having achieved significant trust. This is especially true with the Denial-of-Service (DoS) class of attacks [64]. Attacks like DoS, flooding, Sybil, etc. can result in a total halt of the system's functionality and therefore need to be prevented by addressing the early-stage tactics of an attacker.

Finally, in our design we base the node parameters on those of IoT nodes in a distributed communication network. [65] provides an example of IoT attack detection of more general network attacks but does not take into consideration any form of early detection, whilst [66] uses an early detection approach for IoT devices but does not consider the full range of TAs.

3.3 Current State-of-the-Art and Contribution

Currently, there is no standardised framework of TAs in the literature. Although there is extensive research on the trust inference algorithms themselves [56, 62, 63], the works that consider TAs tend to focus on a small subset of attacks. There have been a number of survey papers that outline the large number of possible TA protocols, [58, 67, 68]. However, all of these are written with the aim of collating the various terminologies and grouping them from a human perspective.

There is a need for an attack framework that combines similar attacks from an ML and detection perspective. [69] begins to outline the learning algorithm's perspective on attacks in a wireless sensor network (WSN). However, it addresses primarily the denial-of-sleep attack rather than accounting for a wider range of possible attacks. Additionally, there are no results provided about the implemen-

tation of the learning algorithm used.

On the ML security side, there are a large number of similarly focused approaches that attempt to characterise a subset of specific attacks in a network. For example, [70] uses a linear SVM to independently detect three different attack protocols in a local area network. However, individually monitoring for each attack would be relatively power-inefficient for a typical IoT battery-operated device [55]. It would be more efficient to develop a detector that can not only flag possible malicious activity but can also then detect which specific strategy is being used. We discuss the full range of TAs further in Section 3.4.2.

The majority of the previously proposed ML detectors use variations on the K-means clustering algorithm [71], which is an unsupervised learning approach to identify clusters or groups within the dataset. Its underlying theory is discussed in Chapter 2.

[72] uses K-means clustering to characterise attacks relating to data packet tampering and showed that the application of ML-based node characterisation improved the overall detection rate of malicious attacks. However, their approach exhibited a minimum 60% error rate for networks made up of 10% malicious nodes or fewer. This is a typical issue for K-means characterisation because honest network nodes generally tend to overlap the attack regions within the parametric space, making it difficult for the detector to differentiate those honest nodes with possible attackers.

A high error rate can also be caused when the K-means algorithm incorrectly identifies the clusters, for example, by assigning zero centroids to small or sparse clusters, whilst assigning multiple centroids within larger, denser cluster. Finally, the K-means approach requires the detector to know in advance exactly how many clusters to divide the data into. This is likely to fail in systems where the malicious

nodes are able to change or adjust their strategy. As already identified, there are hundreds of variations on TA protocols and the detector needs to be able to identify early signs of a wider range of attacks.

The aim of the work reported in this chapter is to provide a full and unified view of the security risks in a trust-based IoT network, and to investigate the benefit of providing human speculative context to ML algorithms in order to more accurately detect malicious nodes prior to attack. As part of this:

1. We identify, through a systematic survey of literature, the attack models that exist in a trust inference network. Based on this, we create a unified framework of network threats from an attack detector's perspective for a trust-based, distributed IoT network.

2. We propose a possible contextual characterisation method for malicious IoT nodes based on the TA framework.

3. We implement a supervised ML algorithm to investigate the effectiveness of our formal attack framework and characterisation for preempting malicious attacks. We also provide a comparison of classification accuracy with the equivalent unsupervised algorithm.

3.4 Creating a Unified Trust Attack Framework

In this section, we will formulate a unified view of TAs on distributed communication networks by collating attack protocols in literature from a ML detector's perspective.

3.4.1 Malicious Node Complexity in Trust Networks

Trust is defined as the probability that the next time one observes a particular node, that node will behave as one expects it to. The trustworthiness, or simply trust, that a node A has about node B is based on a set of previous observations A holds about B, as shown in Fig. 4.3.

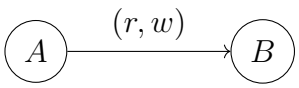


Figure 3.1: Node A has r positive and w negative observations about target B.

As discussed in Chapter 2, the conversion from an observation set to a reputation variable depends on the specific trust inference model used, but many take on a Bayesian statistical approach based on the Beta Reputation Model [73],

$$f(\tau \mid r, w) = \frac{\tau^r (1-\tau)^w}{\beta(r+1, w+1)}, \quad \text{for all } \tau \in [0, 1], \tag{3.1}$$

where τ represents the node's trust, r and w are the positive and negative observations about the node and β represents the Beta distribution.

We begin structuring our attack framework by proposing a broad categorisation of malicious nodes, as shown in Table 3.1.

Tiers 1-3 are differentiated by how aware the malicious node is about the type of trust inference that the network is using to protect itself. The majority of trust inference models can swiftly detect naive malicious nodes, or Tier 1, which are nodes that carry out their attack without consideration or knowledge of the trust inference security in place. For example, a naive malicious node may enter a network and immediately start spreading false information. Its trust rating would quickly decrease and its actions would then have a negligible effect on the network

Table 3.1: Proposed broad categorisation of malicious nodes in a trust inference network

Tier 1: Naive malicious nodes	Attacker commits malicious actions without consideration of how these actions will affect its trust rating nor of how its own trust rating will limit the pervasiveness of its actions.
Tier 2: Passive trust-aware malicious nodes	Attacker follows a protocol that maximises the effect of its malicious actions on the network, for example, by behaving well initially to build up a good reputation.
Tier 3: Adaptive trust-aware malicious nodes	Same as Tier 2 but the malicious node is able to learn about the network's trust inference framework and periodically changes its attack mode to evade detection.

operation. Tier 2 and 3 are more likely to occur in a practical scenario, given that Tier 1 is relatively straightforward to combat using basic security protocols.

It is expected that the *trust-aware* Tiers would typically rely on maximising the malicious node's reputation relative to the honest nodes in the network prior to committing a malicious action. In order to develop this into a more specific framework of attack strategies, we next collate TA models from trust inference literature since 2017 in order to identify the key threats.

3.4.2 Unified Attack Framework

We conducted a survey of trust inference literature since 2017 and observed the trust-aware attacks that have been discussed in each work. As we are looking at these from a prevention and ML viewpoint, it is important to categorise the attacks and identify the key threats that would be of interest to a preventative security algorithm, particularly by merging attacks that differ by only minor variations in

terminology or protocol.

This categorisation is not to be confused with a literature review; given the lack of any standardised view of TAs in existing literature, collating attack protocols and creating this framework is an essential step towards the implementation of ML security in trust networks.

In Table 3.2, we have designated the category *early-stage* attack modes to the four attack protocols: false identity, collusion, on-off and false trust reporting. These are the attack modes that exist only within the trust inference environment because their protocols specifically manipulate the trust ratings within the network in order to maximise the effectiveness, efficiency, or reach of their attack. These attacks would typically precede one or more of the leveraged attacks.

Therefore, by preventing early-stage attacks, the probability of a leveraged attack occurring significantly reduces. We will be focusing our attack framework on signs of early-stage attacks in order to prevent the disruptive part of the attack, such as the 'off' stage in an on-off attack.

Leveraged attacks, in contrast, are those that do not directly involve the manipulation of reputations. They use the attacker's high reputation or pre-existing access to resources in order to have an effect on the network. These include resource-limiting, false data reporting, and recruiting of other nodes.

For example, the false data reporting in Table 3.2 could be the 'off' stage in an on-off attack but, generally, could happen in any network, since they fall under Tier 1 in Table 3.1. The resource-limiting category in Table 3.2 consists of attacks on the network or node resources specifically, which are not dependent on trust but would seem less immediately alarming to a detection algorithm when instigated by a highly-reputed node. Similarly, recruiting another node can also be classed as a leveraged attack because this can be carried out independently of a trust inference

Table 3.2: Unified framework of TAs.

Attack type	Attack name	Protocol	Minor variations and alternative names	With reference to
False identity *(early-stage)*	White-washing	Having committed a malicious action resulting in low trust, the node leaves and re-joins the network with a new identity to reset its reputation.	Newcomer, node replication	[74–86]
	Sybil	A node joins the network under a false identity to evade detection.	False identity	[77, 83–90]
Collusion *(early-stage)*		Two or more nodes work in collaboration to achieve a high trust for at least one of the nodes.	Ballot-stuffing, orchestrated, time-varying attack, wormhole attack	[78, 83, 85, 86, 91–99]
On-off *(early-stage)*		The node behaves normally in the network for a period of time to achieve a high trust ('on' phase) and then executes a malicious action ('off' phase), thereby maximising the effect of that action on the network.	Conflicting behaviour, garnishing, sleeper, camouflage	[74, 75, 82, 83, 87, 91, 92, 97, 100–110]
False trust reporting *(early-stage)*	Slandering	The node C consistently relays bad observations about another node in the network D, resulting in D having a significantly lower trust.	Bad-mouthing, time-varying attack, ballot-stuffing, false validation, defamation	[78, 81, 86, 87, 89–95, 97, 99, 100, 108, 109, 111]
	Self-promotion	The node relays positive observations about itself to other nodes in the network to improve its own reputation.	Selfish attack	[78, 81, 86, 87, 89, 93, 94, 103]
Resource-limiting *(leveraged)*	Denial of service	The node creates a high demand on the network resources such that they become unavailable to other nodes and the network function is disrupted.		[87, 106, 112]
	SSDF Attack	The node sends false information to the channel regulator / base station about whether the channel is in use.		[113–115]
False data reporting *(leveraged)*	Black hole	The node drops all (black hole) or some (grey hole) packets that it receives	Sinkhole, concealment, selective forwarding	[77, 90, 95, 107–110, 116–119]
	Tampering	The node changes existing data without permission.		[77, 81, 87, 90, 95–98, 116, 120]
	Forgery	The node creates new, falsified data.		[77, 81, 87]
	Replay	The node repeats an out-of-date or invalidated packet		[77, 116]
Recruiting other nodes *(leveraged)*	Man-in-the-middle	The node compromises another node and uses its resources or reputation to carry out a malicious action	Eavesdropping	[115, 121, 122]
	Discrimination	Any attack in which the malicious node takes advantage of another node's bad reputation or low trust rating		[81, 83]

network, but having a high reputation would be beneficial to the attacker when misleading honest nodes into cooperating.

The latter strategy takes on various forms in the literature, with a common example being when an attacker A provides false routing information to a node B that has a low trust rating. Given that B, due to its low reputation, would be receiving little information from other nodes in the network, it would be likely to act on the manipulated or falsified data from A. Then, unaware, B would be cooperating with the malicious node A. This is a leveraged attack because, were the attacker A's trust rating low, B would be unlikely to believe the false routing information.

Our framework is collated from a detector's perspective simply by considering what would be required for the attack to have a significant effect on the network. These inferred attack characteristics are highly valuable to a security protocol because they can be used, alongside a ML algorithm, to identify a node as potentially malicious much earlier than the time of attack. We focus on the detection of early-stage attacks since, by preventing these, we would subsequently be able to minimise the risk of leveraged attacks.

3.5 Malicious Node Contextual Characterisation

ML classification algorithms rely heavily on pattern recognition for decision making. However, due to the unclear view of trust network threats, few works in literature have taken a broad ML classification approach to classifying malicious nodes. Based on our proposed framework of TAs in Table 3.2, we can now identify in this section the characteristics of nodes that would be most likely to commit early-stage attacks.

The system, or the observer, can acquire some of the discussed node characteris-

tics based on observation or monitoring of the target node. However, some node properties may need to be read upon entry to the network. An example of this is called anti-cheat [123] and is used to monitor any secondary activity on the device that occurs while the main software is running, with the aim of identifying potential malicious activity i.e., cheating in a multiplayer game network. Similar technology has been picked up by video conferencing programs, and has recently started to be adapted for use in IoT, as outlined in [55].

We now propose a method by which malicious nodes can be characterised based on the attack framework. This contextual information will be used to generate training data for our SVM detector.

Firstly, considering false identity attacks, in which the attacker frequently leaves and re-joins the network in order to refresh its trust rating, we can assume that its network connection length is likely to be shorter than average. Secondly, its trust rating would never reach a stable high given that new nodes tend to be assigned a neutral rating such as 0.5 in most trust models. Therefore, just before an attack, the node is likely to have a relatively low trust rating and will have been in the network for a short period of time.

Similarly, for the on-off attack, the malicious node seeks to achieve a high trust rating as soon as possible to leverage the 'off' phase of its attack. Therefore, it would attempt to communicate with many nodes and with high frequency to raise its trust rating in the eyes of the other nodes. Just before switching to the 'off' phase, it would have achieved a notable increase in its trust rating as the result of interacting with many nodes in a short period of time. A slanderer is likely to follow similar behavioural patterns, but is more likely to seek an even higher trust rating and spend more time in the network building up its trust. Therefore, it is likely that the network connection length of a slanderer would also tend to be

longer.

If a smaller, less powerful node, such as a lightweight IoT device, wishes to maximise its effect on the network, then it is most likely to participate in a collusion-based attack. This would allow it to gain access to wider resources through its colluders, but relies on a high frequency of communication with its colluders.

These attack characteristics are summarised in Table 3.3, with an example given at the end for a public local area network. The specific thresholds would need to be adapted for a particular application and can then be used to form training data for supervised ML algorithms. Note that the example unit of measure for power is million instructions per second (MIPS).

3.6 Support Vector Machine Classification

3.6.1 Methodology

Supervised ML algorithms such as support vector machines (SVM) can utilise labelled training data to identify trends and patterns in new, previously unseen data. In a trust network scenario, the observations that each node holds about other nodes can already be considered as labelled data. Being able to attach more information to the negative observations and attacks, as we have done in the previous section, allows the SVM algorithm to identify other nodes that are likely to commit the same attack in the future.

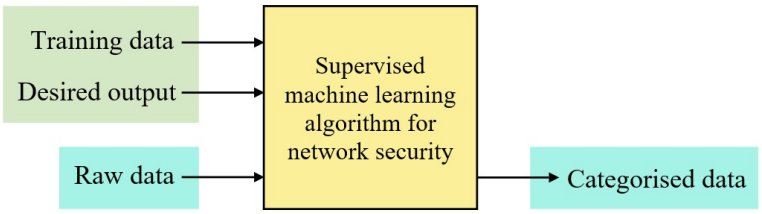


Figure 3.2: Overview of typical ML algorithm input-output.

Table 3.3: Summary of expected attacker characteristics. Example network scenario: ad-hoc, distributed IoT network at a public cafe. Typical nodes would include mobile phones, laptops, street infrastructure nodes such as smart traffic lights, etc.

ATTACK	**Node size** *Example of non-network-specific parameter*	**Node Trust** *Reputation, trust probability, etc.*	**Node Time in Network** *How long the node has been participating in the network*	**Node Reach** *How many others the node has communicated with*
False Identity	-	Low	Short	-
ON-OFF	-	High	-	Frequent communication with a large number of nodes
Collusion	Low	-	-	Frequent communication with a small number of nodes
False Trust Reporting	-	High	Long	Frequent communication with a large number of nodes
Example metric range	**Processing Power** 200-1500 MIPS	**Trust Probability** 0-1	**Time Since Entering Network Range** 0-180 minutes	**Reach in Network** 0-60 transmissions per recipient
Example definition of threshold	Low: <50 High: >120	Low: <0.6 High: >0.9	Short: <40 High: >80	Low: <15 High: >30

An obstacle to implementing this kind of early detection algorithm is that, due to IoT networks not yet being widely implemented, there is little data available about real, past attacks, resulting in a scarcity of real-world SVM training data. Therefore, we statistically generate data sets based on realistic assumptions about the real-world data in order to train our SVM classifier. The process for this is described in Section 3.6.3.

The SVM will observe the labelled training data and use this to identify the various attack regions in the test data. The input is therefore an N by l vector, where N represents the total nodes in the network, and l represents the number of node features being considered, and the output is a vector of length N where each entry is one of the labels *{honest, false ID, on-off, false rep}*.

We have chosen to use an SVM because this method exhibits a higher rate of accuracy when classifying data points nearer to a decision boundary. This was determined by using Matlab's Classification Learner tool to compare the classification accuracy of a range of different ML tools. Being able to classify correctly near to a decision boundary is important because some of the attack regions are closely placed or even overlap, and the detector will need to be able to differentiate between the attacks in this overlap, particularly because honest nodes may also sit within the attack regions. Secondly, given that some of the behavioural regions are likely to overlap or intersect in the parametric space, SVM classification is a better choice of algorithm because it makes use of the kernel trick, which transforms the data into a higher dimension such that it becomes linearly separable.

Realistically, it would be reasonable to expect that honest nodes would occupy the majority of a network, with a few malicious nodes present. The proportion of the network that is honest would likely be higher in a private network than a public network. Let us represent this proportion through the *honest node ratio*, where

$$\text{Honest node ratio} = \frac{\text{Total honest nodes in network}}{\text{Total nodes in network}} \tag{3.2}$$

3.6.2 Support Vector Machine Procedure

The support vector machine algorithm aims to find a hyperplane that separates the data points into clusters on the parametric space. Let X be the vector of data points in the parametric space, then the training data is in the format x_i, y_i, where i_{1-L} represents the data point index, $x \in \mathbb{R}$, and $y = \{-1, +1\}$ represents the vectors from the dividing hyperplane to the two classes. The theory behind the optimisation problem is covered in more detail in Chapter 2, in which the central condition for the linear separation of data is as follows:

$$y_i = (w \cdot x_i + b) - 1 \geq 0. \tag{3.3}$$

The problem is to optimise the margin between support vectors and the decision boundary such that it still satisfies the condition above. The optimal values of w and the constant b are found through updating based on whether the hyperplane successfully divides the two classes. If there is any intersection with either cluster, then $y_i = (w.x_i + b) - 1 < 0$, thus the values are updated.

We have chosen to use a Gaussian kernel because clusters are broadly Gaussian distributed, and although the shape of the clusters in data may vary, the Gaussian distribution is still a closer match than the alternatives, such as linear kernel, sigmoid kernel, etc.

$$K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j) = \exp\left[-\left(\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)\right] \tag{3.4}$$

A Gaussian kernel, as described by (3.4), can be scaled using the σ variable. Let

us call any $\sigma > 1$ as a *coarse* Gaussian kernel, whilst any $\sigma < 1$ will be referred to as *fine*. In Fig. 2.5, we illustrated the effect of the Gaussian kernel on 2D data that is initially linearly inseparable into the two classes present, Class A and Class B. After mapping this data into a higher dimension using the kernel function, the data becomes linearly separable, making classification possible. Our simulation values for three different kernel widths are given in Table 3.4, where the values have been chosen based on an initial simulation - see Fig. 3.7.

Table 3.4: Kernel Scaling Values Used in Simulations

Kernel shape	σ **value**
Coarse Gaussian ($\sigma > 1$)	2
Medium Gaussian ($\sigma = 1$, unscaled)	1
Fine Gaussian ($\sigma < 1$)	$\frac{1}{2}$

The SVM algorithm is designed for binary classification but can be expanded for multiclass classification by performing a broader binary classification between honest and malicious nodes, followed by further binary classifications to identify the 5 classes: honest, false reputation, on-off, collusion and false ID. The resulting block diagram for the SVM classification is given in Fig. 3.3.

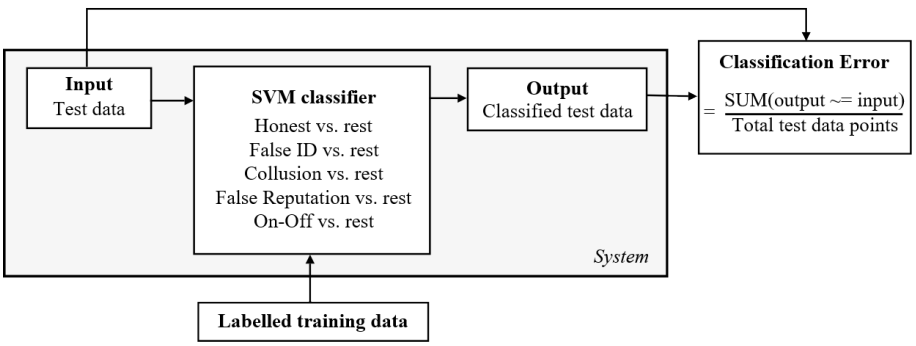


Figure 3.3: SVM classification system block diagram.

3.6.3 Developing a Model for Training Data

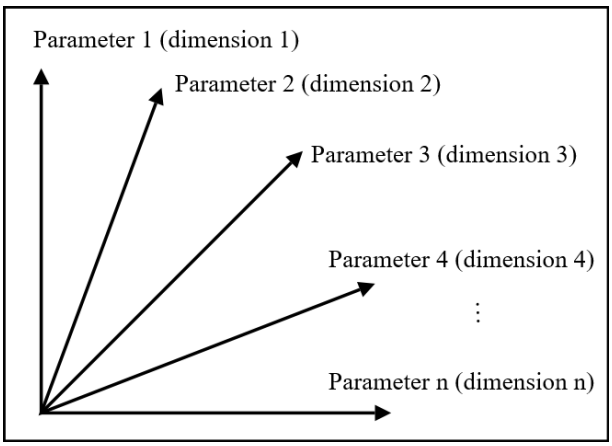


Figure 3.4: n-Dimensional parametric space. In our case, there are four example parameters being used as per Table 3.5: node size, node trust rating, node connection length (time since joining network) and node reach.

Supervised learning algorithms rely on accurately labelled training data in order to classify new data. We present the data within an n-dimensional Euclidean space, where each dimension represents one of the properties of the data being classified, as illustrated by Fig. 3.4. One of the challenges with the implementation of *any* supervised learning in IoT networks is that there is little data available about previous attacks, since IoT networks are yet to be implemented in large enough a scale for there to be sufficient data for training a learning algorithm.

Based on the malicious node characterisation that we proposed in Table 3.5, we can generate realistic training data based on the current view of IoT malicious nodes.

We have characterised malicious nodes by four parameters - their size, reputation, time in the network and reach. Therefore, each node can be represented by a point in a 4-dimensional Euclidean space. Taking the attack regions into account, one would expect the malicious nodes to be Gaussian distributed around each attack

region's centroid to form a 4-dimensional cluster, where the angular displacement of the point from the centroid is assumed to be uniformly distributed.

Let us define an attack cluster as an n-sphere of radius r,

$$S^n = x \in \mathbb{R}^{n+1} : \|x\| = r \tag{3.5}$$

where $n = 4$ given that the parametric space is 4-dimensional. Then we can generate the coordinates of each node in the Euclidean space as follows:

$$x_1 = rcos(\phi_1), \tag{3.6}$$

$$x_2 = rsin(\phi_1)cos(\phi_2), \tag{3.7}$$

$$x_3 = rsin(\phi_1)sin(\phi_2)cos(\phi_3), \tag{3.8}$$

$$x_4 = rsin(\phi_1)sin(\phi_2)sin(\phi_3), \tag{3.9}$$

where r defines the radius of the attack cluster and the uniformly distributed angles $\phi_1, \phi_2 \in [0, \pi]$ and $\phi_3 \in [0, 2\pi]$. Note that the coordinates (x_1, x_2, x_3, x_4) correspond respectively to the normalised parameters: nodes size, node reputation, node time in network, node reach.

In the training data, the nodes lie strictly within their respective clusters. However, the test data clusters have soft margins, meaning that, for example, some of the on-off attack cluster's nodes lie just outside the expected cluster radius r but are still on-off attackers and should be categorised as such. A 3D cross section of the generated data is shown in Fig. 3.5, where the clusters can be clearly seen with additional soft margin points outlined.

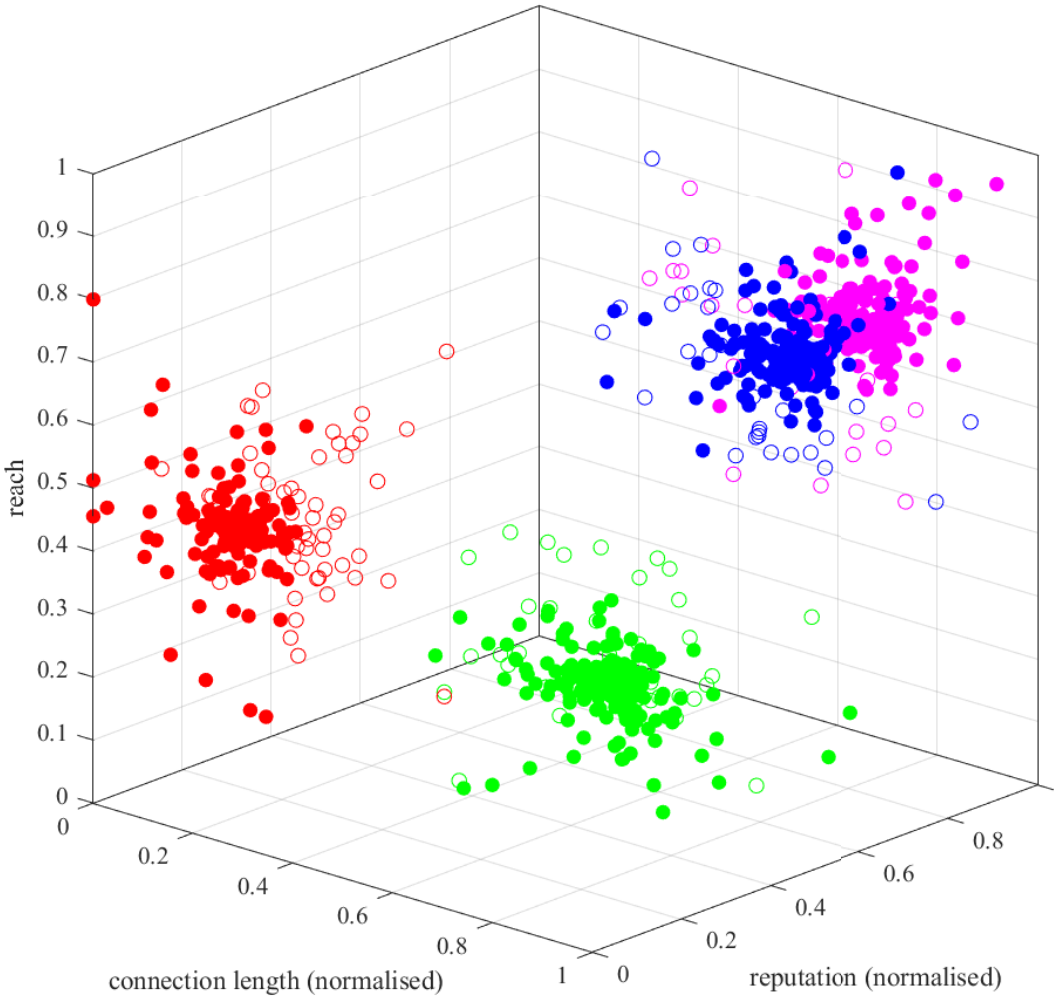


Figure 3.5: Attack data points generated based on the node classification table, for use as training data in a supervised ML algorithm. Each colour represents a different attack mode (red is False ID, blue is On-Off, green is Collusion and magenta is False Reputation Reporting). Filled points represent malicious nodes that lie within their respective attack category margins, whilst empty points represent those that lie outside the attack category margins.

3.6.4 Classification of Largely Malicious Network

We generated training data for each of the attack protocols as shown in Fig. 3.5 and introduced a small number h of honest nodes, which are randomly distributed across the available space. In Fig. 3.6, where the classification accuracy is 90.1%, we can see that the majority of false classifications happen at the edges of clusters, particularly where two attack clusters are in close proximity. It is promising to see from these initial results that the SVM algorithm has been able to identify the majority of the $h = 100$ honest nodes despite their overlap with malicious node clusters.

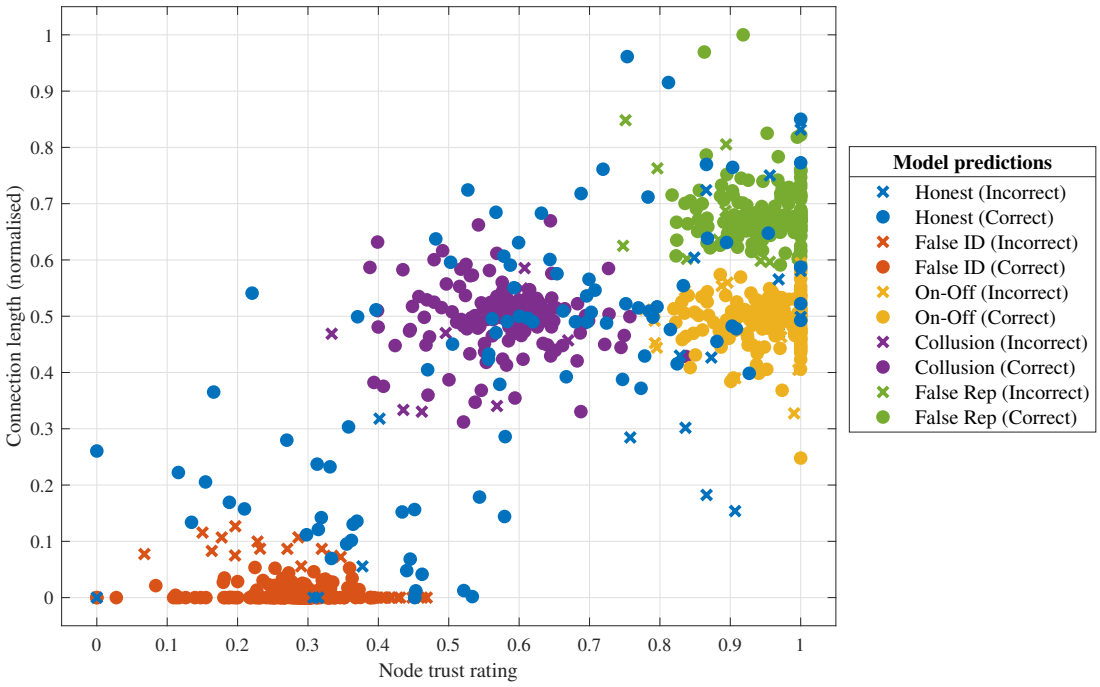


Figure 3.6: Classified test data using the trained SVM for $h = 100$

3.6.5 Gaussian Kernel Function

The SVM classification was run on the training data over a range of kernel scaling coefficients σ - see (3.4). The resulting classification error can be seen in Fig. 3.7, which shows that a sudden increase occurs at $\sigma = 2$, making it a suitable choice for the medium Gaussian kernel.

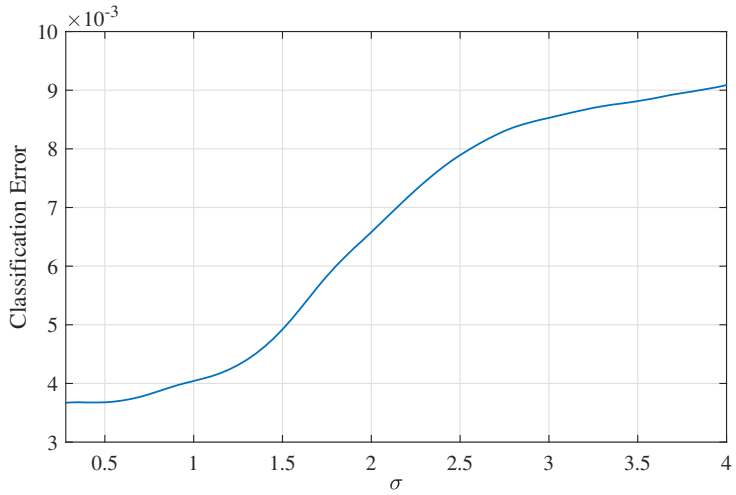


Figure 3.7: SVM classification error for varying kernel scaling coefficients σ.

Typically, $\sigma < 1$ is used for data where different classes are in close proximity in the graph, whilst $\sigma > 1$ is used where the classes have a high rate of overlap. Our training data has both of these properties, so it would be beneficial to not only observe the total error rate but also the types of error occurring in each case. We use $\sigma = \{2, 1, \frac{1}{2}\}$, representing the coarse, medium and fine Gaussian kernels respectively, for which the resulting confusion matrices are shown in Figs. 3.8-3.10 respectively.

Within the total number of misclassified nodes, we can identify three specific error types present in the results, as shown in Table 3.5. In each confusion matrix, the first row represents the nodes that are honest, whilst the first column represents

Table 3.5: Classification Error Types

Error label	**Description**
Misclassification	Any node is misclassified into another class
False alarm	An honest node is classified by the SVM into an attack class
Missed malicious	A malicious node is classified by the SVM as honest.
Misclassified malicious	A malicious node is classified into a different attack class than its own.

the nodes that were classified by the SVM as honest. Interestingly, the coarse Gaussian results in a comparable overall misclassification rate to that of the fine Gaussian, but the coarse kernel resulted in zero missed malicious nodes whilst the fine Gaussian missed 95 malicious nodes. Using the medium Gaussian kernel, where $\sigma = 1$ (unscaled), the classifier resulted in the best overall accuracy of 90.5% but the errors include a mix of missed malicious nodes and false alarms.

False alarms would be the preferred type of error in this scenario since the response of the network can be to simply monitor those nodes that are classified as potentially malicious, rather than closely monitoring all nodes in the network. For example, in the coarse Gaussian results, 85.5% of nodes were correctly classified. Let us say that, in a network of N nodes, M malicious nodes are correctly identified as malicious and there are F false alarms, then the network can more closely monitor a smaller percentage of the nodes, given by (3.10).

$$\% \text{ monitored} = 100 \times \frac{M + F}{N} \tag{3.10}$$

Given that $M \ll N$ in a typical network, this would greatly reduce the number of nodes needing close monitoring. Thus, if one were to choose based on this motivation, then $\sigma \geq 1$ would be a suitable choice.

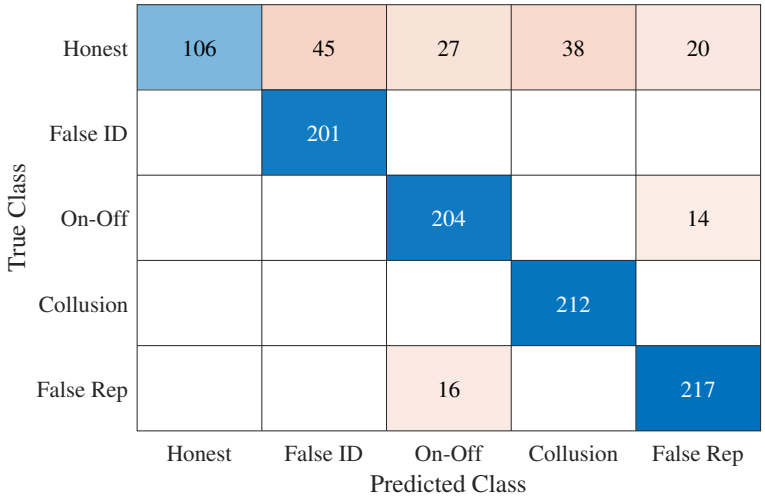


Figure 3.8: Support vector machine classification of the nodes using a coarse Gaussian kernel (scaling factor $\sigma = 2$), with additional honest nodes $h = 100$ and resulting in an overall classification accuracy of 85.5%.

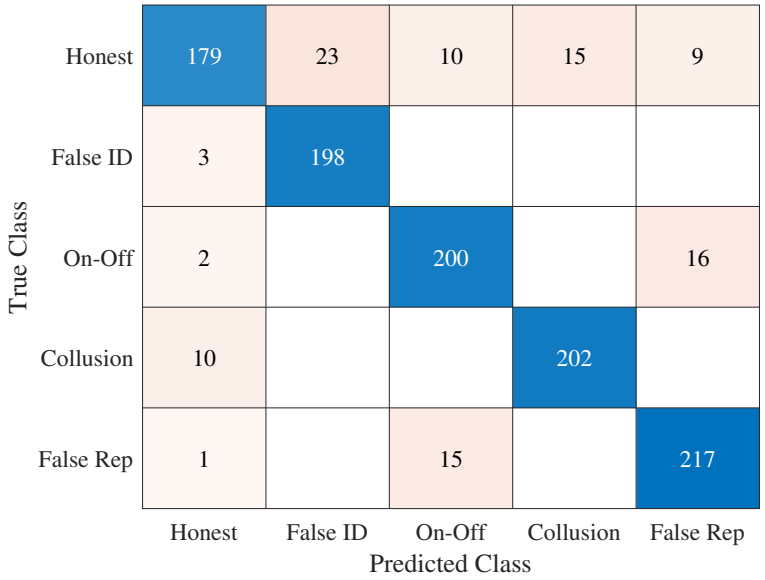


Figure 3.9: Support vector machine classification of the nodes using a medium Gaussian kernel (scaling factor $\sigma = 1$), with additional honest nodes $h = 100$ and resulting in an overall classification accuracy of 90.5%.

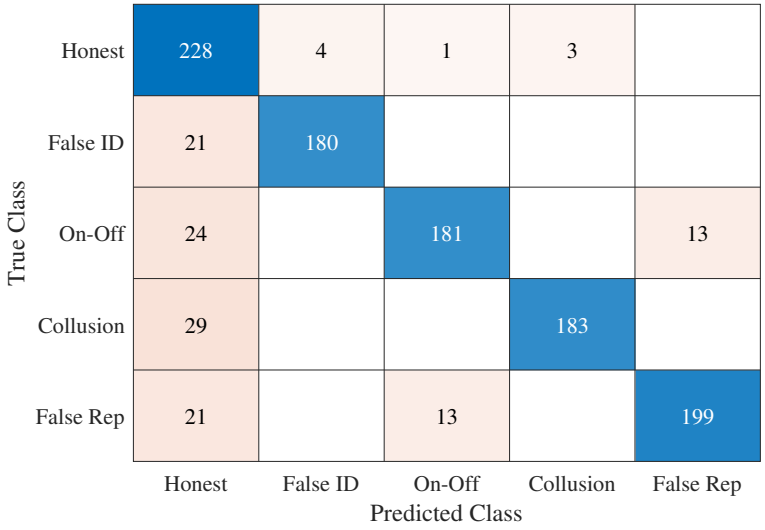


Figure 3.10: Support vector machine classification of the nodes using a fine Gaussian kernel (scaling factor $\sigma = \frac{1}{2}$), with additional honest nodes $h = 100$ and resulting in an overall classification accuracy of 88.3%.

3.6.6 A Largely Honest Network

Now considering the case where $M \ll N$, the detector needs to be able to pick out malicious nodes from a *busy* parametric space that is densely full of honest nodes. An example of the dataset used in this case is given in Fig. 3.11. We then used this to generate training and test data for the SVM, and the results are shown in Fig. 3.12.

The coarse Gaussian kernel again exhibits the highest rate of false alarms and the lowest rate of missed malicious nodes. However, the overall classification error is low for an honest node ratio of 0.9 or less, particularly when using the unscaled Gaussian kernel. In a realistic network, the honest node ratio can reasonably be expected to be 0.8 or above. As expected, the error rate increases abruptly just before an honest node ratio of 1. This is because this region of the graph represents data where there is significant overlap of the honest node region with the attack clusters.

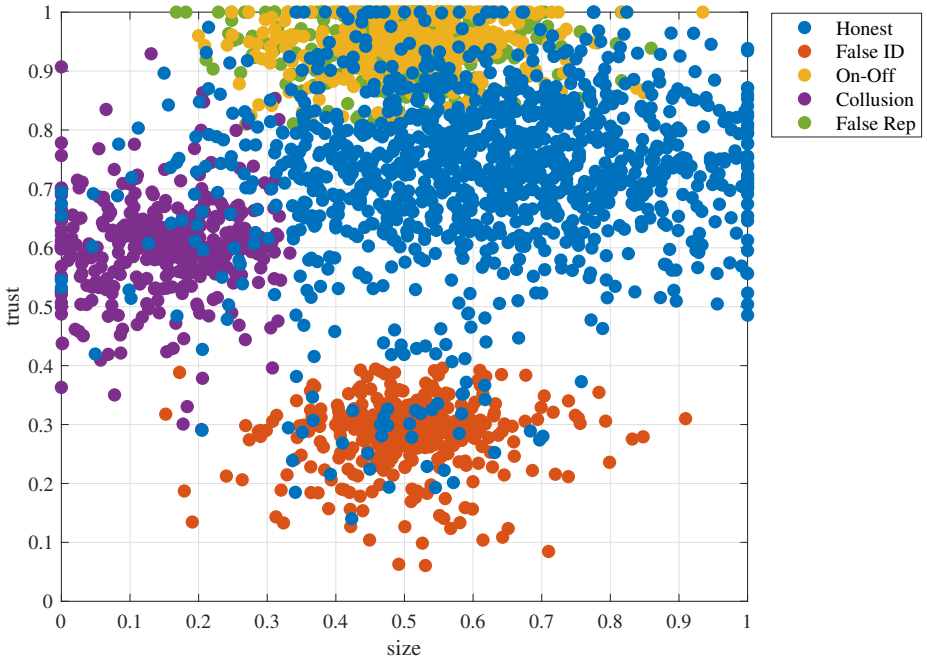


Figure 3.11: Generated data of a network with a higher ratio of honest nodes (high h). The honest node placement is based on the defined average values of each parameter, combined with the assumption that the majority of honest nodes would have a relatively high trust rating.

In order to more closely assess the performance of the classifier in extreme cases of cluster overlap, we next generate data based on the clusters being more broadly distributed across the parametric space.

3.6.7 Severe Overlaps in Node Classes

Some networks, particularly busy public networks, would be expected to have a much broader distribution of honest nodes across the parametric space. Due to the sporadic participation of nodes in the network, some of the node properties, such as connection length, may be shorter. This would also result in a lower average node trust rating given that the node would not have had time to build up a strong reputation.

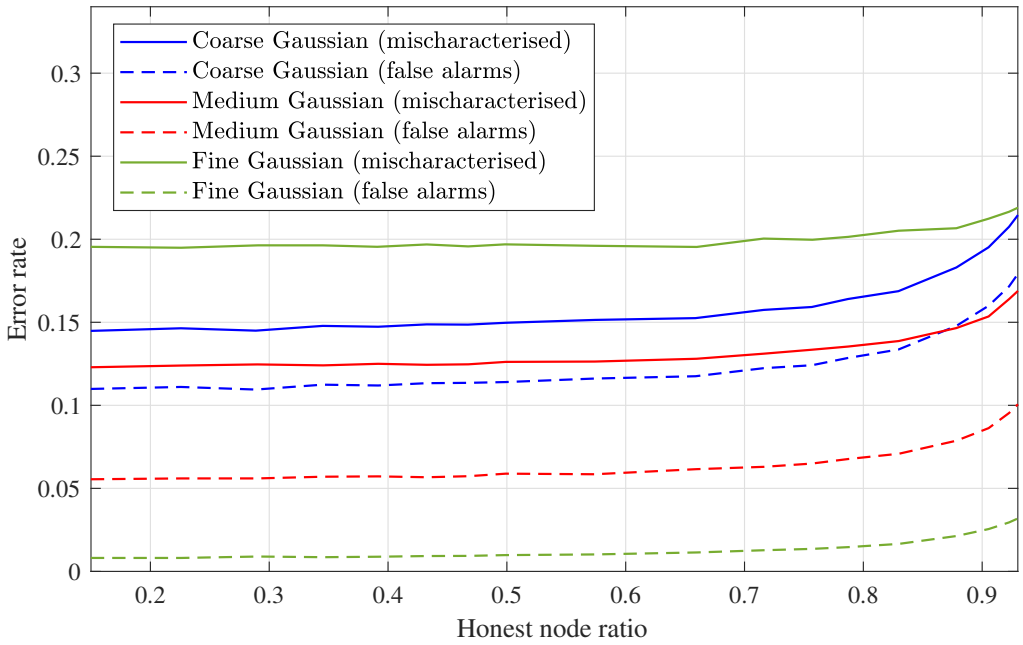


Figure 3.12: Support vector machine classification of the nodes using three different kernels over a range of honest node ratios.

To reflect this, we now simulate an extremely mixed network in which the various classes exhibit a higher standard deviation from the clusters' centroids. Particularly, the honest nodes have been generated with a larger standard deviation in their size, trust, connection length and reach. The generated data can be seen in Fig. 3.13. This was again classified using the coarse, medium and fine Gaussian kernels - the results are presented in Fig. 3.14.

The overall error rates are much higher in this case, as is to be expected given the severity of the overlap between the clusters. The honest nodes densely populate the graph, leaving little room for attack clusters to be recognised as distinct clusters. At an honest node ratio of around 0.9, the medium Gaussian kernel exhibits an error rate of about 30%. This is much higher than its previous 1.5% in Fig. 3.13. However, despite the lack of clarity in the clusters, 70% accuracy with 90% of the network being honest is still an acceptable detection rate, given that a

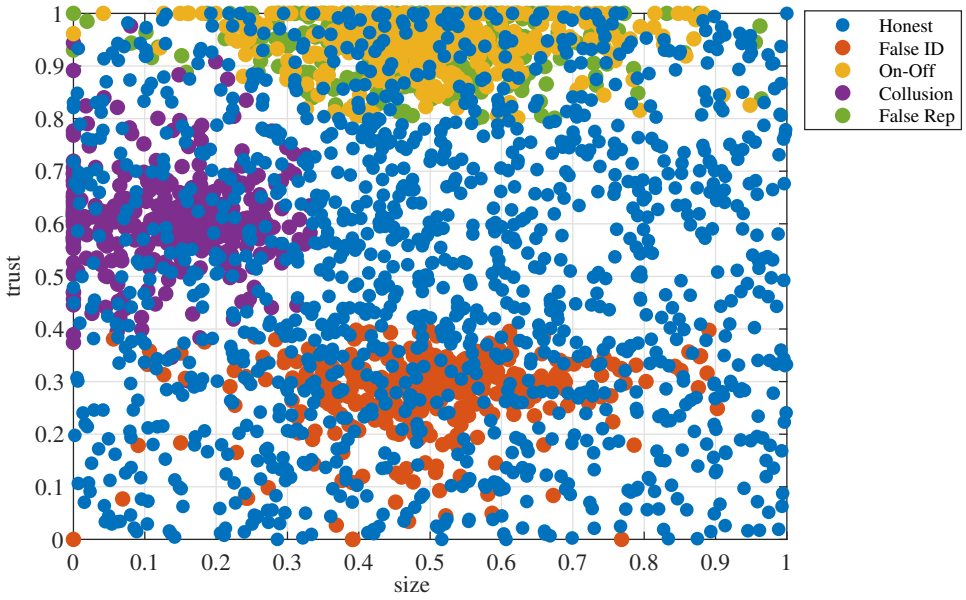


Figure 3.13: Training data model adapted to include densely distributed honest nodes and more widely distributed attack clusters, resulting in a high rate of data overlap between different node classes.

clustering algorithm approach is less likely to successfully determine clusters in this case. Next, we apply a K-means clustering algorithm to similar data sets for comparison with our human-informed SVM approach.

3.6.8 Comparison with Unsupervised Equivalent: K-Means Clustering

So far, we have explored the performance of the SVM classifier when provided with contextual input about where malicious nodes are likely to lie in the parametric space. The advantage of this over an unsupervised equivalent can be investigated by running the same set of test data through an unsupervised K-means clustering algorithm (see Section 3.3). One would expect the K-means approach to produce a similar level of accuracy where the test data is highly clustered, i.e., in the largely malicious network scenario. However, given that the honest nodes

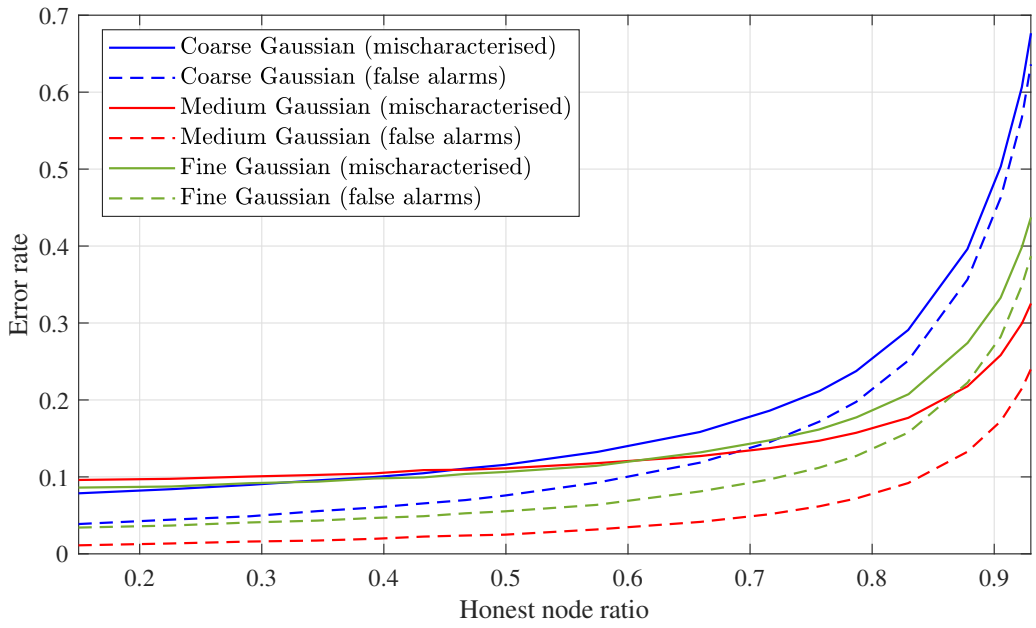


Figure 3.14: Accuracy of the SVM node classification for varying populations of honest nodes within the network, with a comparison of coarse, medium and fine Gaussian Kernel functions.

resemble noise in the parametric space due to their relatively uniform distribution, it is likely that our supervised learning approach will be more successful in a largely honest network, which represents a more realistic network scenario.

Fig. 3.15 illustrates an issue with the K-means clustering algorithm. Clusters with overlapping data points, such as the pink (false reputation) and blue (collusion) attacks, can be mistakenly identified as a single cluster by the classifier. Given that K-means classifiers know how many clusters are present, this subsequently causes another cluster to be assigned multiple centroids, as is the case for the yellow attack cluster (false ID attack). Wrongly assigning centroids then causes the node misclassification rate to increase significantly, given that it affects a minimum of two data clusters each time it occurs.

We ran both, our trained SVM and K-means classification, on the same set of new,

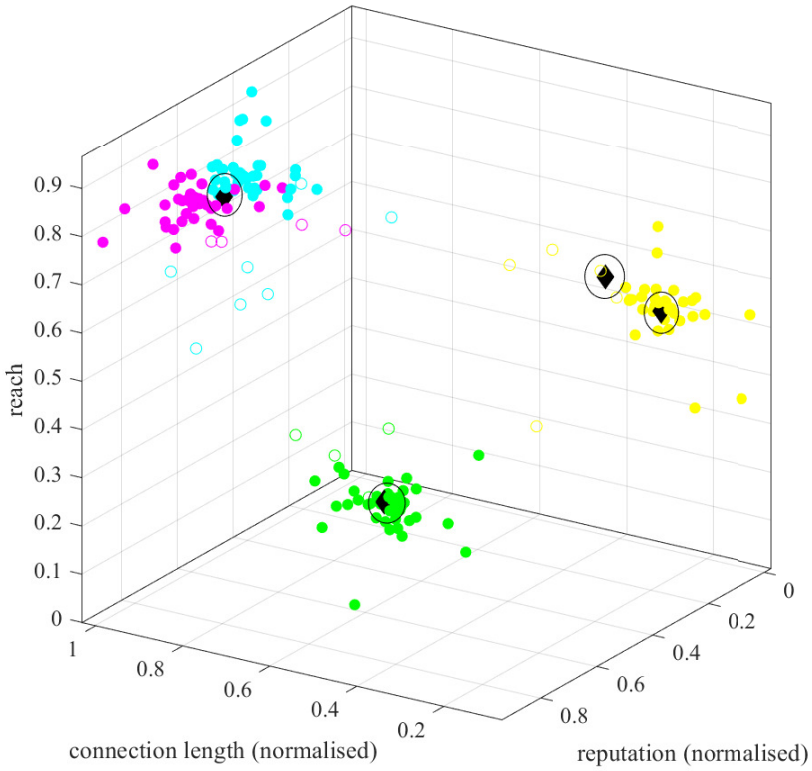


Figure 3.15: The clustering issue when applying the unsupervised K-means clustering method to classify a small number of data points.

unseen data. The results are shown in Fig. 3.16. Although the K-means classifier exhibits a much lower error rate for largely malicious networks, it struggles to identify the attack clusters when the parametric space is densely populated with honest nodes. When the network is made up of 40% honest nodes or more, the SVM provides more accurate classification. Given that a network would typically include around or above 80% honest nodes, we can deduce that the inclusion of the attack characteristics from Table 3.5, based on our unified TA framework, has successfully improved the classification reliability in largely honest networks.

Finally, as a result of the iterative approach used to select centroid locations in K-means clustering, this type of classifier required almost double the execution

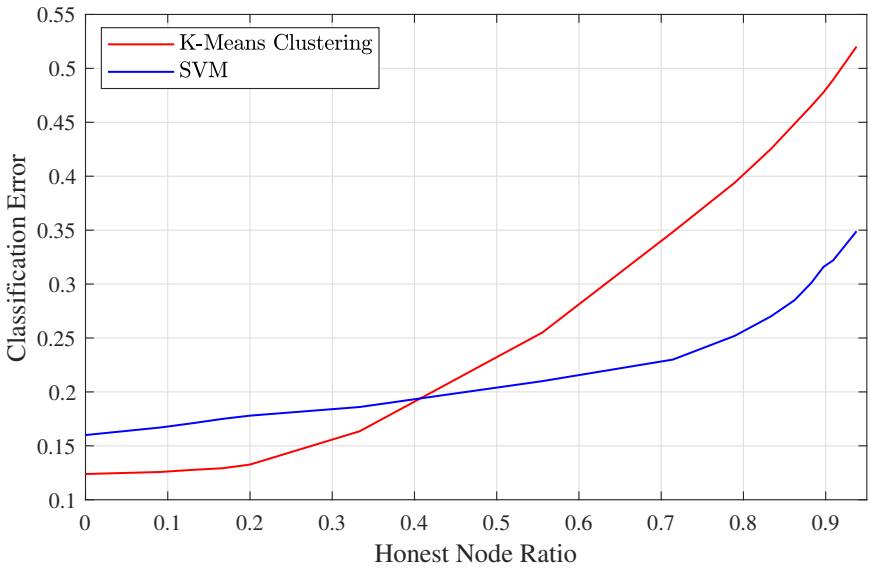


Figure 3.16: Comparison of misclassification rate between unsupervised K-means clustering and supervised SVM. Simulation parameters set to 500 data points per attack, spread (standard deviation) of each data cluster = 0.2, 100 iterations.

time when compared to SVM. This would make it less appropriate than SVM for regular execution by IoT network devices where there is limited power and the computing capacity is low.

3.7 Discussion on Implementation

The novelty of this work lies within its contextual reasoning of malicious node properties, based on a systematic framework that has been designed specifically for the purpose of ML classification of attacks. The parameters we have used are general to the trust-based IoT networks being discussed in current literature, such as the reputation metric, time spent in the network, and node reach. Other, node-specific parameters, and their relative thresholds, can be adapted to fit within the system in which the framework is being applied.

The training data model we developed in Section 3.6.3 provides a useful tool to generate realistic clusters of Gaussian randomly distributed data points, particularly if the parametric centroid of a cluster of points is known. This can be easily expanded for the cases where there is some but not enough attack data to train the algorithm - one can perhaps alter the statistical distribution in our training data model to be centred around the few available points in order to extrapolate more data points from a small number of recorded attacks.

Secondly, we have been able to demonstrate the benefit of combining human speculative intuition as a means of generating labelled training data for ML classification. In comparison, the closest unsupervised alternative, K-means clustering, exhibited low accuracy in classifying the malicious nodes in the presence of many honest nodes: (i) because it was unable to differentiate between clusters where clusters where closely placed to each other, and (ii) because the honest nodes were consistently misclassified by the K-means clustering into their closest attack cluster. In contrast, the kernel trick has allowed the SVM algorithm to differentiate honest nodes from a cluster of malicious nodes, despite there being a significant overlap in some cases.

Clearly, the use of an SVM at all in the network will add some level of processing and complexity to the system. However, the implementation of an SVM is highly adaptable, with the main cause of any latency being the training of the SVM itself. Based on a network's needs, the designer would need to consider a trade-off between the SVM accuracy and network overhead by tuning factors such as: (i) How often does the SVM need to be re-trained? (ii) Should the SVM sacrifice its accuracy through the use of broader kernels and smaller datasets in order to maintain low-latency decisions? Our results showed that the majority of erroneous classifications are overly careful, i.e., by classifying honest nodes as potential threats. Even with a low accuracy SVM, our proposed approach can sup-

ply a powerful tool for understanding which network nodes should be monitored, rather than using the resources to monitor all network nodes.

Finally, IoT networks are relatively new in practice [55]. Despite a clear need for AI-based distributed security for IoT, there will likely be little or no data available about real-world IoT TAs until those networks have been in wider use for a few years. In the meanwhile, before the training data is available, the logical reasoning of malicious node characteristics in Table 3.5 could prove to be a powerful tool for the early detection of attacks in dense WSNs, public ad-hoc networks and so on.

3.8 Conclusion

In this chapter, we have proposed a unified framework of trust network attacks based on collating attacks from literature from an AI detection viewpoint. We used this framework to develop a malicious node characterisation method, based on the likely behaviours of a malicious node prior to an attack. This has been done with a view to achieving early detection of malicious activity, in order to take on a preventative approach rather than managing consequences after attacks.

We then used our malicious node characterisation to generate realistic training data for a supervised ML algorithm, SVM, and used it to classify simulated IoT network data to detect TAs. The SVM classifier exhibited a much higher classification accuracy when compared to its unsupervised equivalent, K-means clustering, which was unable to accurately classify nodes in networks made up of 40% or more honest nodes. The SVM protocol resulted in high classification accuracy when classifying potentially malicious nodes, including when the test data included soft margins with significant overlap between attack clusters. However, the SVM presented a higher misclassification rate when the parametric space was

populated by a high number of uniformly and densely distributed honest nodes.

As a result of this work, we have achieved a comprehensive and logical representation of the possible attacks in trust networks. Now that we have explored how to identify and classify potential malicious nodes in a network, the next steps would be to understand how to leverage this knowledge to reinforce the network security.

4 | Trust-Based Data Management Protocol for a Distributed Communication Network

Abstract

In this chapter, we propose a novel data management, distribution and aggregation protocol, which utilises trust information about nodes to maximise the accuracy of that data. The aim of our proposed protocol is to enable a fully distributed network, comprised of limited-capability nodes, to function without external intervention. Our design utilises elements from distributed ledger technology, whilst adapting them to be within the scope of implementation for typical communication devices. In conjunction with this, the proposed trust-based data aggregation protocol takes advantage of input from untrustworthy nodes to reinforce the aggregate data, which is in contrast with comparable work in the literature that would instead attenuate or exclude malicious node participation.

4.1 Introduction

Existing communication systems are largely centralized, not only in their architecture but also in terms of their data exchange, resource allocation and network management protocols. Owing to the rapidly growing number of connected devices in the world, it is hoped that distributed networks will be the basis for much of our future communications due to their scalability, flexibility and scope for automation, such as in vehicle-to-vehicle networks [124].

One of the biggest challenges in the implementation of distributed communication networks lies in reconfiguring network protocols to operate in a correspondingly distributed manner, without reliance on a central authority or hub. Taking the Internet of Things (IoT) as an example, an ideal IoT smart home network should function with limited human input and should be able to protect itself from network attacks. This and related expectations for IoT networks are outlined further in [125].

In Chapter 3, we explored the use of trust inference for distributed network security, and demonstrated that trust information about nodes can be used to identify and classify attacks on trust networks. In this Chapter, we investigate what this would look like in terms of distributed protocol design, and how trust can be integrated into a communication network to facilitate secure data exchange and consensus.

So far, the distributed network self-management and security protocols proposed in literature tend to be significantly more computationally intensive than their centralized counterparts, making them impractical for networks involving typical connected devices such as sensors, mobile phones, laptops, automated vehicles. A notable example of this in practice is distributed ledger technology (DLT), such

as blockchain (BC) networks. BC networks, first proposed in [126], provide a strong premise for distributed security; their transactions are anonymized, peer-to-peer, and are validated by other participants in the network using an encryption puzzle called proof-of-work. However, the proof-of-work algorithm is so computationally intensive that it can only be solved by a small minority of devices in the network, and additionally incurs a high transactional latency in the realm of 5-10 minutes [127]. Meanwhile, the average BC network node relies on those very few capable nodes to function securely.

Similarly, other examples of distributed network protocols seem to delegate crucial network functionality to a supposedly trustworthy minority of nodes, participants or third-party intervention, such as cloud computing. It cannot be assumed that this option will always be available to a network, such as in wireless sensor networks (WSNs), where it is not uncommon for sections of the network to become 'separated' for periods of time due to highly variable channels.

4.1.1 The Problem and the Proposed Solution

There is a need for a self-management framework for secure, timely data exchange in a distributed network, without relying on high complexity validation techniques in order to function.

This problem can be separated into two specific needs:

1. Although BC networks provide a suitable starting point for distributed data management, they would not be suitable for the majority of networks in practice. This is because frequent communications or a large number of connected nodes would result in large quantities of data being stored at each node. Therefore, in our approach, we use elements of the BC protocol but we restructure its data storage and exchange procedure to retain the dis-

tributed nature of data handling in BC, but with a significantly reduced the overhead on network nodes.

2. As part of the self-management of the network, the nodes need to be able to evaluate the reliability of the data without hefty computational overheads such as Proof-of-Work (PoW). A particular challenge in distributed networks is that multiple nodes may hold different versions of similar or the same data with varying validity (age of the version), corruption (damage or dropped bits during transmission) and accuracy (is the node being honest?). Therefore, we accompany our proposed data management scheme with a novel, trust-based data aggregation protocol. This will utilise the nodes' past experiences of each other in order to reinforce the outcome, thereby enabling them to more independently make informed decisions about the reliability of each other and the data.

4.1.2 Existing Work

In existing literature, related data management protocols tend to specifically delegate tasks based on the nodes' individual abilities. For example, Ethereum *light nodes* [128] cannot contribute to decisions, but participate in a BC network by storing only the block headers, which they receive from *full nodes* [129]. Similarly, the existing literature on lightweight distributed ledgers for IoT systems relies on the assumption that there will always be inherently trustworthy and capable nodes available to act as the full nodes, but this is not necessarily the case in every network. Note that the term *lightweight* is broadly used in literature to describe a protocol or system that is suited to typical mobile devices.

In our data aggregation model, the nodes measure the reliability of any received data by estimating the trustworthiness of its source. Related aggregation protocols

LDAT and RDAT [130] also use Bayesian trust inference in lightweight networks, but these are designed to reduce the network overhead on the channel rather than on the individual nodes. Within multi-sensor data fusion, direct aggregation is usually performed using maximum a priori (MAP) estimation, a survey of these is provided in [131], and MAP estimation in existing literature is largely based on extensive prior knowledge about the data rather than about the sources of the data, which is not always readily available to all nodes. In a related distributed security framework, IOTA Tangle [132], the submitter of a transaction is required to first validate two older network transactions before it may commit its own transactions to the waiting list. However, a malicious node may take advantage of this by validating transactions arbitrarily in order to participate in the network. Additionally, this approach is particularly susceptible to collusion-based attacks, because it results in transactions being approved by relative newcomers in the network, rather than longer-term participants with a track record of reliability. Tangle requires each transaction to be supported by a PoW [126] action, which is too computationally complex to be completed by typical IoT nodes.

4.1.3 Contributions

The contributions of this work are as follows:

1. We propose a secure, scalable distributed ledger storage protocol for a distributed network with limited-capability nodes.

2. We propose and analyze a trust-based data aggregation scheme, which aims to improve the reliability of information sharing in trust networks. Unlike existing trust-based methods, the proposed method utilises interactions with nodes that are known to be untrustworthy in order to reinforce the aggregation.

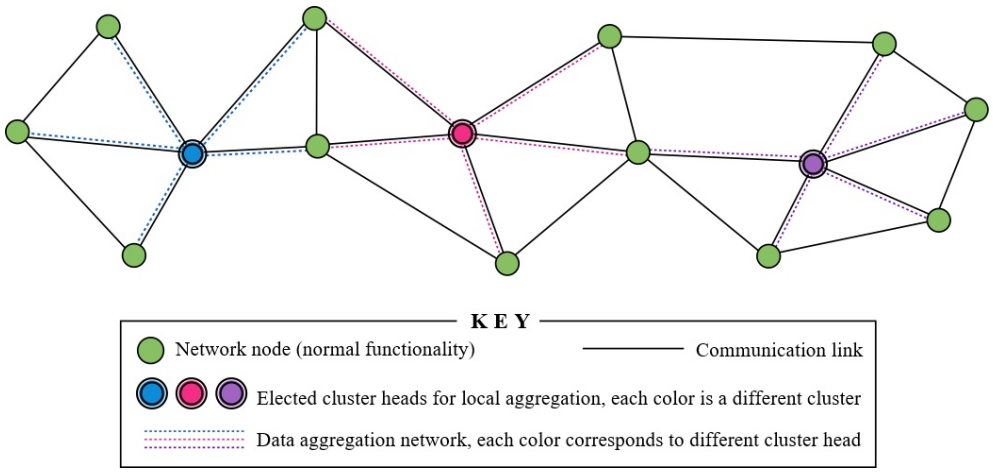


Figure 4.1: Example of the WSN cluster showing the communication links vs. aggregation clustering.

3. We expand our data aggregation protocol to multi-bit data and discuss the practicality of its application.

4. We investigate the asymptotic behaviour of the trust-based aggregation scheme for large and small networks.

4.2 Distributed Data Management Protocol

4.2.1 System Description

In this section, we propose our novel distributed data management protocol using elements of BC technology that have been adapted for common networks, and which don't rely on access to powerful computing resources. The protocol is presented with respect to an example WSN environment.

Let there be a WSN cluster of N nodes. The primary function of the wider network is to collect geo-specific binary data X about its environment, where each cluster is responsible for one particular area, as shown in Fig. 4.1.

Every node i in the network includes a sensor, which periodically measures $X \in \{0, 1\}$ in its locality and sends this data as a report $Y \in \{0, 1\}$ to other nodes in the network. It is assumed that each node i believes its measurement of X to be true, but that it is possible to have $Y \neq X$ in two cases:

1. If the reporting node is dishonest, it would choose to report a value of Y that is different from what it believes to be the true data, i.e., if it believes $X = 1$ then it would choose $Y_i = 0$, and vice versa.

2. If an unreliable channel results in the report becoming distorted.

4.2.2 Data Storage and Exchange

A distributed ledger is maintained to keep a time-stamped and immutable record of sensor readings. There is one cluster head (CH) as chosen by the other nodes as the result of a periodic, trust-based Raft [24] consensus election. The CH has only two responsibilities: (i) packaging data into blocks (mining), a process that is checked by the remaining blocks, and (ii) broadcasting data requests from its followers.

The sensors collect some binary data from their environment, for example, motion sensors recording 1 when motion has been detected since the last update and 0 otherwise. These nodes send their readings to the CH but retain a copy temporarily. The CH ceases its sensing for the period of its leadership, and instead mines the other nodes' readings into fixed-size blocks, which are then distributed randomly. Each of the contributing sensor nodes checks for the inclusion of original reading within the mined block using a Bloom filter [133]. These nodes then update their respective trust value for the CH by recording the mining interaction as a positive or negative observation of the CH. Periodically, a Raft [24] election is called to elect a new CH (miner) based on updated trust values.

4.2.3 Novel Lightweight Storage Scheme

The distributed ledger is collectively maintained by the network as a record-keeping mechanism. The data is mined into chronologically interlinked blocks of fixed size. In our example scenario, this could include all aggregated sensor measurements over a specific period of time. However, unlike typical BC implementations that require all nodes to store a copy of the full ledger, in our proposed model the nodes store a *random subset* of the blocks, whilst still ensuring that each block is stored in a distributed fashion across a subset of network nodes.

An example of this is illustrated in Fig. 4.2, where each node uses 25% less storage space for blocks than if it were to store the full network ledger. However, each block is still stored as a copy at the majority of network nodes. This is only a small cluster, but when the same principle is applied to larger networks, it can significantly reduce the amount of data stored at each node whilst maintaining the distributed nature of each block's storage.

The allocation of blocks to nodes is done on a random basis to reduce the probability of malicious collusion. Whenever a node needs to access data that it doesn't store locally, it requests this data from the network using the proposed data retrieval protocol in Section 4.3. Any nodes storing that particular block will send a copy to the requester, who then aggregates the many versions of the data together based on its trust values for the respective senders. The aggregation method is presented in Section 4.4.

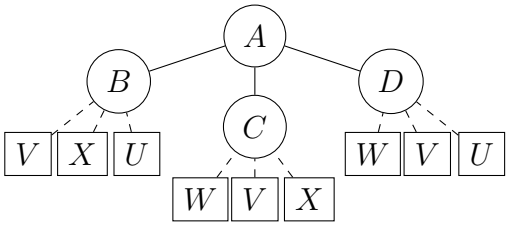


Figure 4.2: Cluster of nodes after a Raft election, showing that A is chosen to take on the miner role. The solid links show the flow of data while the dashed links show data storage. This is also a small-scale illustration of how the random distribution of blocks U, V, W and X means that each node only has to store 75% of the total data, whilst each block is still stored at multiple locations.

4.3 Data Retrieval

4.3.1 Observation Model

Monetary networks such as BitCoin [126] rely on balances and transaction history to validate new transactions. IoT networks cannot validate new contributions in the same way, because sensor readings and other scientific data can vary independently from past data. We instead use trust inference to assess the validity of new submissions to the network ledger, given that data from a highly trustworthy source is more likely to be valid.

The trust information about each node is utilised in the data aggregation scheme in Section 4.4, with input from both trustworthy and untrustworthy nodes being used to strengthen the aggregation result. Secondly, in the CH elections, each node votes for the candidate for which it holds the highest trust rating. This increases the probability that the most honest and least error-prone node is chosen.

The trust metric is based on a set of observations recorded by the nodes about each other. With reference to Fig. 4.3, trust is defined as the probability τ that D will behave as C expects it to, and is a function of C's positive r and negative w

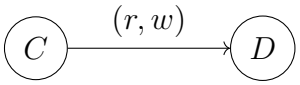


Figure 4.3: Node C has r positive and w negative observations about target D.

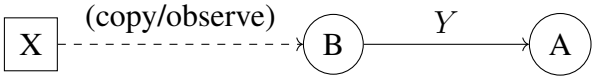


Figure 4.4: Node B retrieves original data X and reports it as Y to node A.

observations about D.

A *set of observations* is achieved over time as nodes record new observations whenever they interact with each other. For example, after checking a mined block or after data aggregation. The observations are then converted into a numerical representation of trust using, for example, the Beta Reputation Model [73] as described below.

4.3.2 Beta Reputation Model

Note that we base our trust calculation on the beta approach, but the novelty of the proposed aggregation scheme in Section 4.4.2 comes from exploiting the output of trust inference rather than from generating the trust inference results. Thus, the aggregation can be applied in conjunction with results from *any* trust inference model.

Consider two nodes, A and B in Fig. 4.4, in a WSN. A requests data block X from the network via the CH. B locally stores X, so it sends a copy of X to A in the form of a report Y. Ideally, the report accurately reflects the original data $Y = X$, where $X \in \{0, 1\}$. Depending on its intention, B may have lied by inverting the data ($Y \neq X$). The probability that B will tell the truth is represented by its trust value τ. According to the Beta Reputation Model [73], τ's probability density

function can be written as

$$f(\tau \mid r, w) = \frac{\tau^r (1-\tau)^w}{B(r+1, w+1)}, \quad \text{for all } \tau \in [0,1] \tag{4.1}$$

where $B(a, b)$ represents the beta function. However, it is the nature of the beta function that one cannot know exactly the value of τ. Instead, τ's probability density function is used to find the expected value of trust $\hat{\tau}$.

4.3.3 Effective Trust

For the purpose of data aggregation, we extend the BRM to account for the channel error rate e, which is the probability that data is inverted during transmission from B to A due to a poor communication channel, regardless of B's intention. For reference, $e = 0.05$ in the BitCoin BC [126].

As a result of e, any observations made by nodes in the network are not directly indicative of the sender's trust τ but rather of its *effective trustworthiness*, which accounts for both τ and e. We define effective trust as $p = \mathbb{P}(Y = X)$, which is given by

$$p = \tau(1-e) + e(1-\tau). \tag{4.2}$$

and, like τ, has a beta distribution meaning that we cannot know exactly the value of p but can estimate it by using observations,

$$\hat{p} = \frac{r+1}{r+w+2}. \tag{4.3}$$

The relationship between the original data X and a report about the data Y can then be represented as the binary symmetric channel (BSC) in Fig. 4.5. Note that $\hat{p} = 0.5$ should be used for newly discovered nodes for which there are no

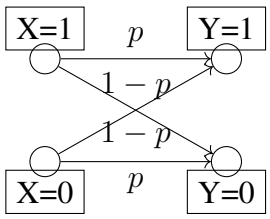


Figure 4.5: Binary symmetric channel for the report Y about data X from any single source node.

prior observations ($r = w = 0$) in order to ensure that their effect on the data aggregation is minimal, given that data from both effectively trustworthy ($\hat{p} > 0.5$) and untrustworthy ($\hat{p} < 0.5$) source nodes is used to reinforce the result.

4.4 Data Management

4.4.1 Data Retrieval

In the distributed storage proposed in Section 4.2.3, nodes store a subset of the blocks from the network ledger. This aims to reduce the storage overhead on each node, but if a node requires some data that it does not locally store then it may need to retrieve this data by requesting the associated block from other nodes in the network.

Assume in Fig. 4.2 that a node D requires a particular data point in block X. For simplicity, let us assume that X contains a single-bit, binary data point of interest. D does not store this block locally and so it needs to retrieve copies of it from source nodes, which are nodes that store the desired block X. D sends a request to the CH, A, where the request contains the identifier of the desired block X. A then broadcasts a request for the block along with the identifier of the requester. B and C, acting as source nodes, then respond to D with their versions of the desired data in the form of reports Y_B and Y_C. Finally, D uses its values for $\hat{p}_B$

and $\hat{p}_C$ to aggregate the two reports into a *best guess* $\hat{X}$ for the true value of X, i.e. trust-based data aggregation.

4.4.2 Trust-Based Data Aggregation Scheme

We propose here a method by which the requesting node can aggregate all of the data reports by finding the most probable value of X based on the Y and τ values of each source node. Let X be a single, binary data point of interest. Some prior information is known about the data, $s = \mathbb{P}(X = 1)$. It is possible to find the marginal probability of the report $\mathbb{P}(Y)$ by applying the law of total probability to the BSC in Fig. 4.5. From this, the likelihood $\mathbb{P}(Y|X)$ can be calculated using Bayes' theorem. The system in Fig. 4.4 can then be expanded to include multiple source nodes $n > 1$, which each store local versions of X. A makes a request for the data X and receives n versions of this data in the form of reports $\boldsymbol{Y} = [Y_1, Y_2, \ldots, Y_n]$. This problem can be represented as the maximum a posteriori (MAP) estimation shown in (4.4) where $x \in \{0, 1\}$.

$$\hat{X} = \arg\max_x \mathbb{P}(X = x|\boldsymbol{Y}), \tag{4.4}$$

where $\mathbb{P}(X = x|\boldsymbol{Y})$ can be written using Bayesian probability as

$$\mathbb{P}(X|\boldsymbol{Y}) = \frac{\mathbb{P}(\boldsymbol{Y}|X)\mathbb{P}(X)}{\mathbb{P}(\boldsymbol{Y})}. \tag{4.5}$$

Re-writing this with the assumption that each node behaves independently results in the following relationship, which can be substituted into (4.4) to produce the aggregate result:

$$\mathbb{P}(X|\boldsymbol{Y}) = \frac{\mathbb{P}(X)\prod_{i=1}^{n}\mathbb{P}(Y_i|X)}{P(\boldsymbol{Y})}. \tag{4.6}$$

4.4.3 Aggregation Error Rate

An aggregate output from the system is considered accurate when it equals the input, therefore, for a dataset with L total bits,

$$E = \frac{\sum_{b=1}^{L} \left(\hat{X}_b - X_b \right)}{L}. \tag{4.7}$$

This is the aggregation error rate, or E, which measures the performance of the proposed data aggregation scheme. However, there are multiple avenues through which an error can occur, as noted in the binary symmetrical channel in Section 4.3.3.

Let E_1 and E_0 represent the two cases of error, $\mathbb{P}(\hat{X} \neq 1 | X = 1)$ and $\mathbb{P}(\hat{X} \neq 0 | X = 0)$ respectively, then

$$E = \mathbb{P}(\hat{X} \neq X) = E_0(1 - s) + E_1 s. \tag{4.8}$$

Proposition 1. *Let there be a constant k such that*

$$k = \frac{n}{2} + \frac{1}{2} \frac{\ln \frac{1-s}{s}}{\ln \frac{1-\hat{p}}{\hat{p}}}, \tag{4.9}$$

then the conditional probabilities of error E_0 and E_1 can be calculated for any n using (4.10) and (4.11).

$$E_0 = \sum_{i=0}^{\lfloor k \rfloor} \binom{n}{i} (1 - p)^{n-i} p^i, \tag{4.10}$$

$$E_1 = \sum_{i=\lceil k \rceil}^{n} \binom{n}{i} (1-p)^i p^{n-i}. \tag{4.11}$$

The above is only true for homogeneous trust, i.e. where every source node participating in the network has the same value of expected trustworthiness. Note that this does not necessarily mean that they are all equally trustworthy, but that they have the same probability of being trustworthy.

It can then be shown, as follows, that the aggregation error rate E is a function of the total number of zeros reported. Let $Z = \sum_{i=1}^{n} \mathbf{1}_{\{0\}}(Y_i)$, where Z is a binomial random variable and $\mathbf{1}_{\{0\}}(x) = 1$ for $x = 0$. Therefore, we can estimate the probability density function of Z by using a Gaussian approximation of the binomial distribution.

Proof. The aggregation result is defined for a single report using a MAP calculation,

$$\hat{X} \;=\; \arg\max_x \frac{\mathbb{P}(X = x, Y)}{P(Y)} \;=\; \arg\max_x \left[\frac{\mathbb{P}(X = x)\mathbb{P}(Y|X = x)}{\mathbb{P}(Y)} \right]. \tag{4.12}$$

Expanding for n source nodes gives

$$\hat{X} = \arg\max_x \left[\frac{\mathbb{P}(X = x) \prod_{i=1}^{n} \mathbb{P}(Y_i|X = x)}{\mathbb{P}(Y)} \right]. \tag{4.13}$$

The error case E_0 is defined as $\mathbb{P}(\hat{X} \neq 0 \mid X = 0)$, we can rewrite the MAP decision in terms of $\hat{X}$,

$$E_0 = \mathbb{P}\left[\frac{\mathbb{P}(X=1)\prod_{i=1}^{n}\mathbb{P}(Y_i|X=1)}{\mathbb{P}(Y)} > \frac{\mathbb{P}(X=0)\prod_{i=1}^{n}\mathbb{P}(Y_i|X=0)}{\mathbb{P}(Y)} \mid X=0\right]. \quad (4.14)$$

The aggregation error rate E is a function of the total number of reports that equal zero Z. Substituting in $s = \mathbb{P}(X=1)$, $\hat{p} = \mathbb{P}(Y_i = X)$, and $Z = \sum_{i=1}^{n} \mathbf{1}_{\{0\}}(Y_i)$,

$$\begin{aligned} E_0 &= \mathbb{P}\left[(1-\hat{p})^Z \hat{p}^{n-Z} s \right. \\ &\qquad \left. > \hat{p}^Z (1-\hat{p})^{n-Z}(1-s) \mid X=0\right] \\ &= \mathbb{P}\left[(2Z-n)\ln(1-\hat{p}) + (n-2Z)\ln(\hat{p})\right. \\ &\qquad \left. > \ln(1-s) - \ln(s) \mid X=0\right] \\ &= \mathbb{P}\left[Z < \frac{n}{2} + \frac{1}{2}\frac{\ln\frac{1-s}{s}}{\ln\frac{1-\hat{p}}{\hat{p}}} \mid X=0\right]. \end{aligned} \quad (4.15)$$

Note the sign change based on the assumption that $0.5 < p < 1$, because $p < 0.5$ would not be a suitable operating range for a network. Following similar steps for the remaining error case $E_1 = \mathbb{P}(\hat{X} \neq 1 \mid X = 1)$,

$$\begin{aligned} E_1 &= \mathbb{P}\left[\frac{\mathbb{P}(X=0)\prod_{i=1}^{n}\mathbb{P}(Y_i|X=0)}{\mathbb{P}(Y)} \right. \\ &\qquad \left. > \frac{\mathbb{P}(X=1)\prod_{i=1}^{n}\mathbb{P}(Y_i|X=1)}{\mathbb{P}(Y)} \mid X=1\right] \\ &= \mathbb{P}\left[Z > \frac{n}{2} + \frac{1}{2}\frac{\ln\frac{1-s}{s}}{\ln\frac{1-\hat{p}}{\hat{p}}} \mid X=1\right]. \end{aligned} \quad (4.16)$$

□

Note that Z is a sum of Bernoulli random variables in $\boldsymbol{Y}$, with a distribution based

on p and s. As a result, Z becomes a binomial random variable $Z \sim \mathcal{B}(n, p)$ for $X = 0$ and $Z \sim \mathcal{B}(n, 1-p)$ for $X = 1$, then we arrive at (4.10) and (4.11).

We can presume that the trustworthiness of nodes will be quite high in some cases, for example, in private IoT networks where trust is used to measure the reliability of nodes. We can observe this aggregation for such networks by approximating its performance for p near to 1. To do this, (4.15) and (4.16) can be expanded and approximated as shown in (4.17),

$$E_0 = \binom{n}{\lfloor k \rfloor}(1-p)^{n-\lfloor k \rfloor} + \mathcal{O}\left((1-p)^{n-\lfloor k \rfloor+1}\right). \tag{4.17}$$

We can also approximate E_1 near $p = 1$ as

$$E_1 = \binom{n}{\lceil k \rceil}(1-p)^{\lceil k \rceil} + \mathcal{O}\left((1-p)^{\lceil k \rceil+1}\right). \tag{4.18}$$

Having considered small networks, we next analyse E for large n, as would be the case in a distributed ledger network or some WSNs. This allows us to assess the scalability of the proposed scheme.

Proposition 2. *Let k be defined as in Proposition 1, then for large n the conditional probabilities of error, E_0 and E_1, are well approximated by (4.19) and (4.20) respectively:*

$$E_0 \approx \frac{1}{2}\left[\operatorname{erf}\left(\frac{n-np}{\sigma\sqrt{2}}\right) - \operatorname{erf}\left(\frac{\lceil k \rceil - np}{\sigma\sqrt{2}}\right)\right], \tag{4.19}$$

$$E_1 \approx \frac{1}{2}\left[1 - \operatorname{erf}\left(\frac{\lfloor k \rfloor - n(n-p)}{\sigma\sqrt{2}}\right)\right]. \tag{4.20}$$

Proof. Z can be approximated using the Gaussian distribution $Z \sim \mathcal{N}(\mu, \sigma)$,

where

$$\mu = \begin{cases} n(1-p), & \text{if } X = 1 \\ np, & \text{if } X = 0 \end{cases} \tag{4.21}$$

$$\sigma^2 = np(1-p). \tag{4.22}$$

We then find the cumulative distribution function of Z for large networks by applying the Central Limit Theorem. □

Based on the above, we can then consider scenarios with extremely large n by analysing the error functions in (4.19) and (4.20) for $n \to \infty$.

$$E_0 \approx \frac{1}{2}\left[\left(\frac{e^{-{z_2}^2}}{z_2\sqrt{\pi}}\right) - \left(\frac{e^{-{z_1}^2}}{z_1\sqrt{\pi}}\right)\right] \tag{4.23}$$

$$E_1 = \frac{e^{-{z_3}^2}}{2z_3\sqrt{\pi}} \tag{4.24}$$

Where $z_1 = \frac{n-np}{\sigma\sqrt{2}}$, $z_2 = \frac{\lceil k \rceil - np}{\sigma\sqrt{2}}$ and $z_3 = \frac{\lfloor k \rfloor - n(n-p)}{\sigma\sqrt{2}}$. Given that $\lim_{n\to\infty}(\lceil k \rceil, \lfloor k \rfloor) \simeq \frac{n}{2}$, and letting $a = \frac{\frac{1}{2}-p}{\sqrt{2p(1-p)}}$, we can approximate both types of error:

$$E_0, E_1 \simeq \frac{e^{-a^2 n}}{2a\sqrt{n\pi}}. \tag{4.25}$$

4.4.4 Multi-Bit Binary Data

In a direct extension of the aggregation to multi-bit data, the aggregator would seek to solve the MAP problem:

$$\hat{\boldsymbol{X}} = \arg\max_{\boldsymbol{x}} \mathbb{P}(\boldsymbol{X} = \boldsymbol{x} | \boldsymbol{Y}), \tag{4.26}$$

$$\mathbb{P}(\boldsymbol{X}|\boldsymbol{Y}) = \frac{\mathbb{P}(\boldsymbol{X}) \prod_{i=1}^{n} \mathbb{P}(\boldsymbol{Y_i}|\boldsymbol{X})}{P(\boldsymbol{Y})}, \tag{4.27}$$

where $\boldsymbol{X}$ is a multi-bit data packet of length L bits, $\boldsymbol{x}$ represents one out of 2^L different possible values, and the reports are in the form $\boldsymbol{Y} = [\boldsymbol{Y_1}, \boldsymbol{Y_2}, ..., \boldsymbol{Y_N}]$, where each node i's multi-bit report $\boldsymbol{Y_i} = [y_{i1}, y_{i2}, ..., y_{iL}]$. However, this approach would result in significantly high processing times. As a result, this is not a sustainable approach, and multi-bit data would need to be aggregated in another way.

Instead, a simple and effective solution would be to treat each bit independently and apply the proposed aggregation in (4.6) to each bit in turn. This is a straightforward implementation for most systems. However, it is not necessary for the whole packet to be aggregated if only a subset of the bits require aggregation.

For example, let us consider a typical packet transmission set-up, in which data is encoded, modulated and transmitted at one end of an additive white Gaussian noise (AWGN) channel, and then received, demodulated, and decoded at the other, as shown in Fig. 4.6.

A low-density parity check (LDPC) [134] decoder could be used to easily and quickly identify the bits within each set of packets $\boldsymbol{Y}$ that require aggregation. LDPC is a common choice of error checking protocol, and can itself correct errors in packet-based data. Importantly, it can also identify the positions of erroneous bits in data, thereby requiring aggregation only on those bits.

Another variable is the timing of the attack or error. There are two likely positions along the transmission timeline where the data can be affected. This can either occur at the source of the data (Attacker Model 1), or within the communication channel (Attacker Model 2), as shown in Fig. 4.6. Model 1 is likely to occur with dishonest network participants sending incorrect data, whilst Model 2 is likely to

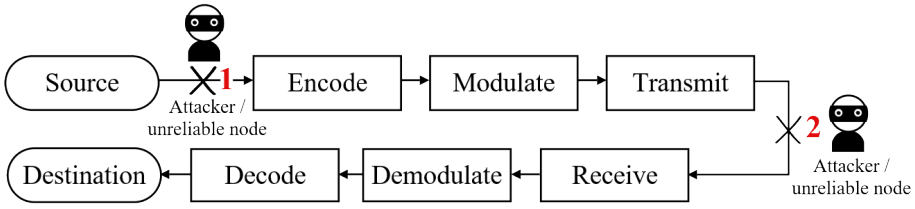


Figure 4.6: Wireless transmission block diagram showing presence of unreliable or malicious node at two different positions. position 1 indicates that the data is affected prior to encoding, whereas position 2 indicates that the data is affected whilst in the channel.

occur as the result of a poor channel or a third-party attack, such as the Man-in-the-Middle attack [135].

Secondly, one would need to consider the specific actions of malicious nodes. For example, a malicious node could invert all bits in the data, send old data or attack each bit with an independent probability of attack. These will each result in different proportions of bits in the packet being flipped.

Whether the packet can successfully be recovered depends on both of the points above, the timing of the attack with respect to the transmission sequence, and the extent of the damage to the packet. For example, Model 2 attacks would affect not only the data bits in the packet but also the LDPC code bits, meaning that severe damage from a Model 2 attack would result in a higher probability that the packet data is unrecoverable. Therefore, it may be beneficial to perform the aggregation *prior* to decoding in order to also recover the code bits via aggregation.

A summary of the possible attack scenarios is given in Table 4.1.

Table 4.1: Overview of Packet Attack Scenarios

Attack Scenario	**Description**	**Solution**
Correctable	The LDPC decoder is able to identify the damaged bits.	LDPC decoder can rectify most or all of the packet, followed by aggregation across all reported packets..
Detectable	The LDPC is able to identify that a packet has been attacked but not which bits.	Perform aggregation after the decoder in order to recover the damaged bits.
Undetectable	The attacked LDPC codeword resembles a valid codeword: the decoder is unable to detect an error.	Perform aggregation before decoding to recover the LDPC codeword.

4.5 Numerical Results

In this section, we will present the simulation methodology and pseudocode for the simulation of the proposed trust-based aggregation protocol, followed by numerical results.

4.5.1 Simulation Methodology

A MATLAB Monte Carlo simulation is used to observe the aggregation error rate E. We define the prior distribution of data $s = 0.5$ and the channel error probability $e = 0.05$.

In the first simulation, we find E against the number of source nodes n for different p values using (4.19) and (4.20). For comparison, an alternative, commonly used method called majority selection [136] has also been simulated, which chooses $\hat{X}$ using the statistical mode of many binary reports Y. In subsequent simulations, we randomly generate the data X and each node's corresponding report Y based on the node's respective p value. It is assumed that the receiver holds a $\hat{p}$ value for each source node which is, for the simulation, equal to p. Using Equations (4.15) and (4.16), we can then perform the MAP estimation and find the aggregation error rate E by comparing the MAP result with the original data.

We simulate a homogeneous network in which all nodes have the same expected trustworthiness p, but this does not mean that the nodes are all equally trustworthy. The trust probabilities of the nodes in a use-case scenario can be generated using any existing trust inference model.

For the multi-bit data where the error occurs after encoding, the code structure for the simulation is given in Algorithm 3 and the Matlab code snippet for the LDPC configuration is given in Appendix E.1.

This specific configuration of the LDPC encoder used is given in Appendix E, along with the underlying pseudocode for the multi-bit simulations. This is only one example of a parity check matrix, and is the default 3/4 quasi-cyclic parity check matrix in Matlab's LDPC function, resulting in a packet length of $L = 486$ bits. However, any parity check matrix can be used depending on desired L, block length, and redundancy.

4.5.2 Results

Fig. 4.7 shows a comparison between using the proposed trust-based aggregation method and choosing the result based on a majority vote. The proposed method demonstrates significantly lower single-bit error rate for all $n \geqslant 2$ showing that, even for a small number of source nodes, integrating trust inference greatly improves the aggregation reliability.

Fig. 4.8 shows the relationship between the aggregation error rate E and the nodes' effective trustworthiness p for single-bit data. The error rate is much lower as the number of participating source nodes n increases. The simulation agrees with the binomial calculation of E and the normal approximation provides a close representation of the curve. The graph also shows that the asymptotic analysis is accurate to the simulation, and thus provides a reliable method by which to

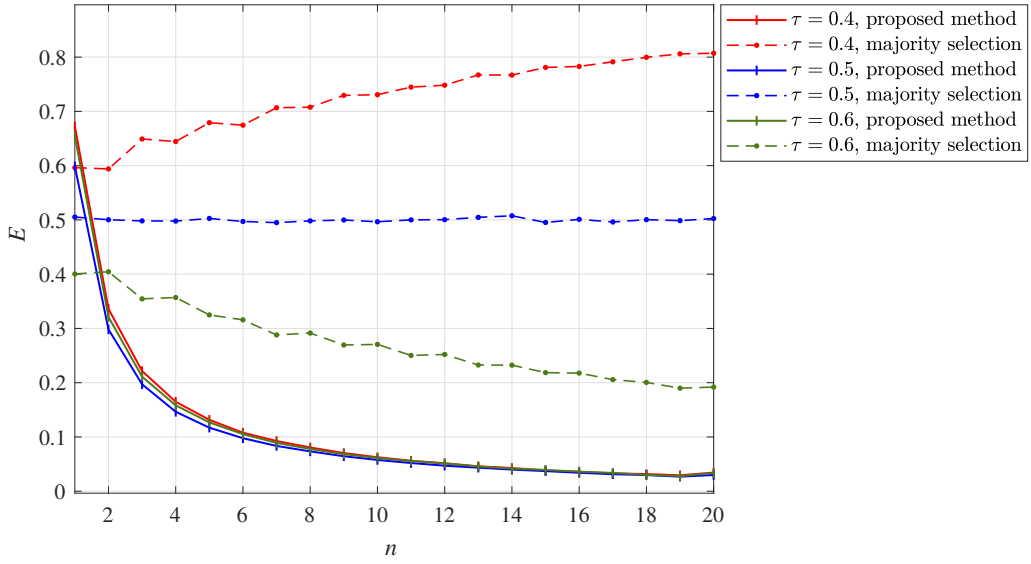


Figure 4.7: Single-bit error rate E vs. source nodes n for the proposed method and the majority selection method for various values of node trustworthiness τ.

observe the system for large n, which represents extremely large networks.

As the aggregation utilises input from both trustworthy and untrustworthy nodes, the results are symmetrical about $p = 0$. However, we focus here on $0.5 < p < 1$ because this would be a more suitable range of trust for a network to operate in. E peaks at $p = 0.5$ because this represents a point where the trust information is not useful, such as in new networks. The maximum E corresponds to $\min\{s, 1 - s\}$ because, when there is insufficient data about the trust and channel (in the form of p), then the aggregator chooses the most likely value of X based on the prior information.

Fig. 4.9 shows that the proposed scheme exhibits low single-bit error rates even for very small networks in the range $n \leq 10$, and the approximation around p close to 1 follows the simulation results closely. This is particularly promising because real-world networks are likely to lie in this region of the x-axis, where,

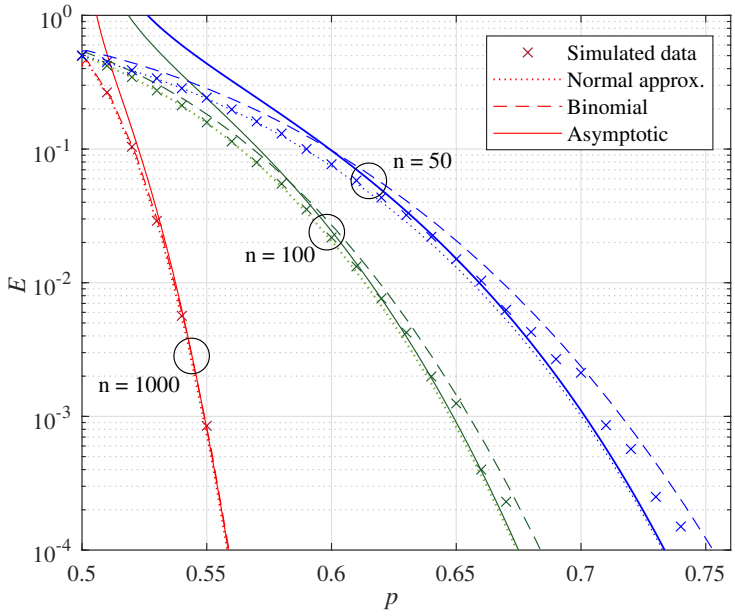


Figure 4.8: Comparison between the simulation, binomial calculation, normal approximation and asymptotic analysis of E for different scales of n and for single-bit data.

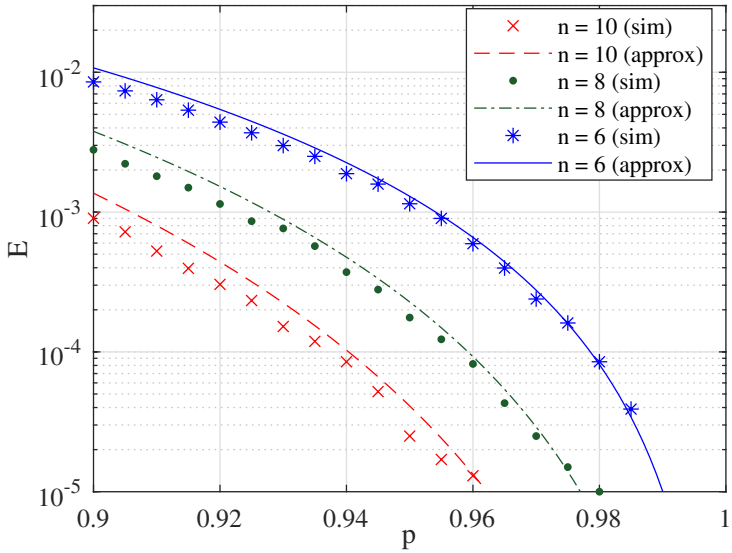


Figure 4.9: Simulation and approximation of aggregation error rate in small networks for p close to 1 for single-bit data.

on average, network nodes are likely to be trustworthy.

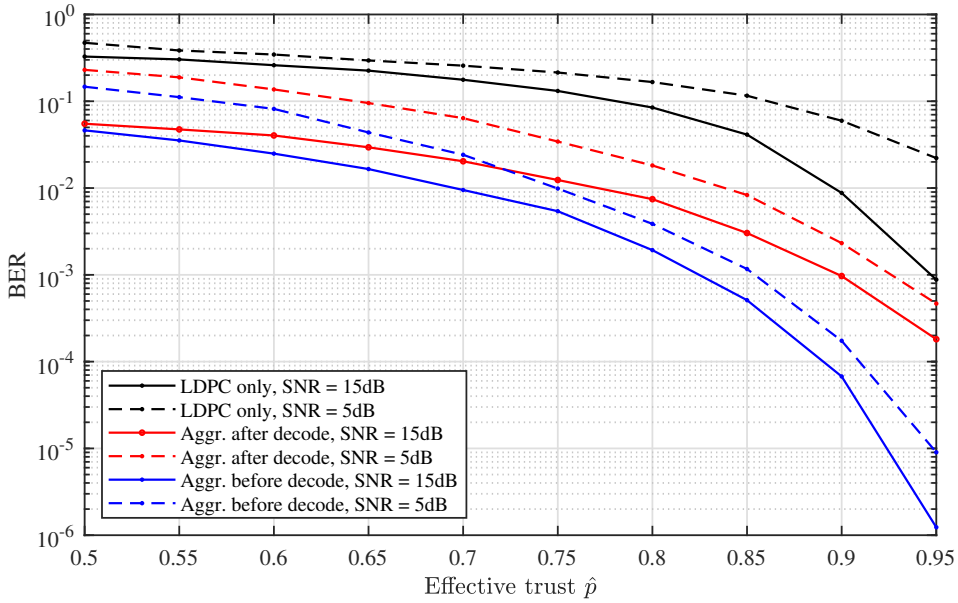


Figure 4.10: BER over a range of $\hat{p}$ values. Simulated for packet-based transmission of multi-bit data with $SNR = 20dB$, $L = 486, n = 20$. Comparison given of performing the aggregation before the LDPC decoder versus after.

In Fig. 4.10, we simulated a multi-bit data transmission based on Fig. 4.6, with both types of attacker present. The results show that our proposed trust-based aggregation presents an immediate improvement in bit error rate (BER) versus using LDPC alone. Naturally, a lower BER is observed when operating in higher signal-to-noise ratio (SNR). Aggregation before decoding results in a lower BER, which is likely due to the aggregator being able to recover some of the LDPC code bits before they are passed to the decoder.

We then simulated the scenario where each node is independently affected by error or an attacker. In this case, fig. 4.11 shows how the availability of source nodes n affects the BER of the system using an LDPC in conjunction with the bit-by-bit trust-based aggregator. A typical network would be expected to have $p \approx 0.9$, for which a larger n results in a steep drop-off in BER. However, as seen in 4.12, the choice of n results in a trade-off between the BER and the aggregation time per

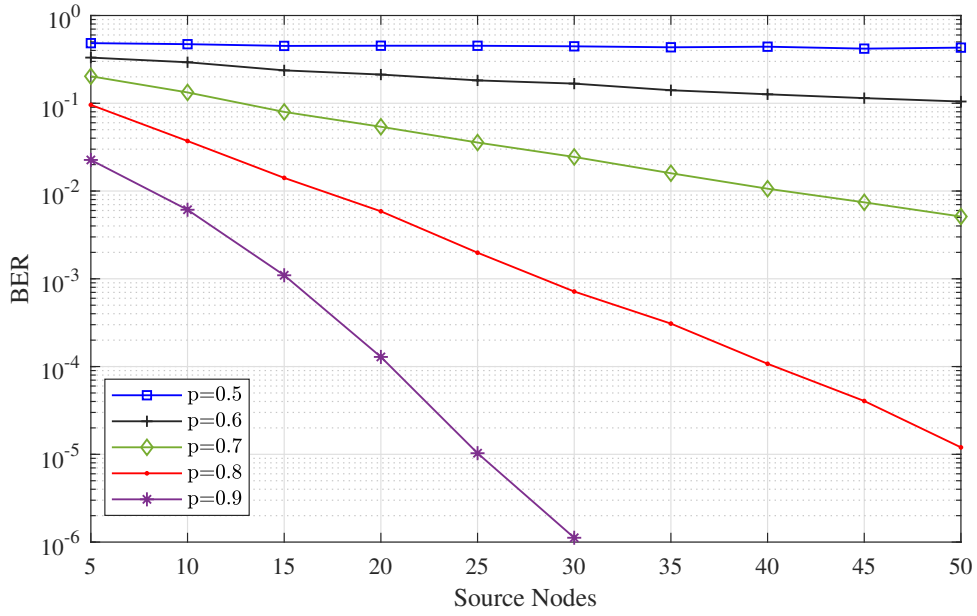


Figure 4.11: Multi-bit BER vs. n for LDPC with aggregation after decoding. $SNR = 20dB$, $L = 486$.

packet, the latter of which is measured by setting a timer within the simulation that begins just before the aggregation and terminates afterwards. In terms of implementation, the network could self-organise based on smaller clusters in order to facilitate faster data processing, or instead larger clusters with lower BER and higher aggregation latency.

4.6 Discussion on Implementation

This work focuses on a fully distributed network where there are no trustworthy, highly capable nodes present to take on storage, processing and computational overheads. For example, a WSN deployed in a very remote region or community networks in rural areas. Secondly, typical distributed network devices, such as IoT sensors, have intermittent connectivity as a result of low battery life, poor

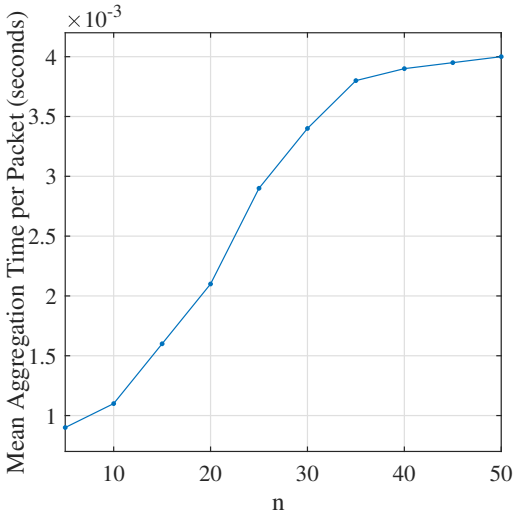


Figure 4.12: Time taken to reach a resolution on the aggregation for each packet vs. number of source nodes n involve in the aggregation. $SNR = 20dB$, $L = 486$, $p = 0.7$.

communication channel or malicious attacks. These limitations mean that, even in a network that originally had access to more capable full nodes, the network may lose access to these for periods of time. Were this to happen, the network would need to function temporarily as a fully lightweight IoT network, perhaps using our proposed lightweight distributed ledger and data aggregation schemes to continue functioning until the more capable nodes are restored. As a result of the ledger, there would still be a complete record of all interactions during the network down time. Surprisingly, little work has been done to address this scenario, and the closest protocol is the naive majority selection scheme [136] or simple MAP-based selection, both covered in Section 4.5.2.

Further use cases exist within the wider field of multi-sensor data fusion, such as photographic image stitching by WSNs in low visibility regions. [137] describes how this is a particular challenge in adverse weather such as sandstorms due to noisy images and intermittent connections, and our proposed data management

and aggregation protocols would be ideal for accurately aligning the image overlaps between neighbouring nodes. Note that an all-encompassing channel error rate constant e has been used in this research, which is sufficient here for the purposes of investigating our proposed protocols. However, it may be beneficial in future to apply a specific channel model to reflect different use cases, such as a high level of multipath fading in smart city networks, or severe Doppler shift in drone networks.

The data aggregation itself can also be used for a wide variety of purposes. On service platforms, such as eBay [138], the overall reputation score for a user is typically calculated as a mean of all ratings. However, a common issue is that different users have different views on what constitutes a positive interaction. For example, some sellers always rate buyers as positive, regardless of the true nature of the interaction, specifically in the hope that they will receive a favourable rating in return. This is a common occurrence across a variety of trust-based service and e-commerce platforms, such as Uber, Vinted and Deliveroo [139]. Using a trust-based score aggregation such as ours would effectively normalize the ratings being given, with respect to the rater's bias.

Although the distributed data management protocol used elements of DLT, to be directly applied to a crypto-network, we would need to integrate additional protocols for the division of the encryption overheads between multiple nodes. Delegation protocols have been discussed in literature [140]. In most DLT and BC implementations, each of the N network nodes stores every block from the network ledger. The percentage reduction in storage requirement per node due to our proposed approach would be proportional to $1 - n_{min}/N$ where n_{min} is the minimum number of source nodes required to achieve some maximum allowable error rate E_{max}.

4.7 Conclusion

In this chapter, we have presented a novel data self-management protocol for distributed networks with limited computational resources. To facilitate the operation of this framework, we also designed and presented a trust-based data aggregation protocol, which successfully provides a means for network nodes to independently evaluate each other and any data received, by observing other nodes and determining their effective trustworthiness.

Simulations results show that the proposed method successfully utilises reports from untrustworthy nodes to reinforce the aggregate data output and thereby reduce the error rate. Higher error rates were seen around $p = 0.5$, but this isn't usually a concern for networks because this would only occur where the network is relatively young with nodes holding limited observations about each other, or where the network is extremely mixed such that the network-wide $p \approx 0.5$. However, the simulation results also show that the aggregation error rate is extremely low in networks with a mean effective trust across the nodes of $p \approx 0.9$, which is reasonably representative of a typical network.

For multi-bit data, the simulation results showed that the aggregation protocol could provide a highly reliable means of combining input from multiple sources. It was also shown that, in conjunction with LDPC, multi-bit aggregation is relatively straightforward to implement without significant rise in network overheads. However, this could be further improved by determining the size of the network clusters, n, specifically to balance the trade-off between BER and aggregation execution time.

Chapter 3 explored how to detect and classify trust attacks and this chapter has demonstrated how that knowledge can be used to facilitate secure self-management

and consensus between distributed network nodes. A particular question remains: what about new networks? We have discussed in Chapter 2 how distributed networks are particularly vulnerable to new nodes, since little trust information is available about them. The next chapter focuses on the classification of nodes in this early period by restructuring the trust data into a hidden Markov model.

5 | Hidden Markov Model Approach for the Early Classification of Node Behaviour

Abstract

We have previously identified that trust networks are particularly vulnerable to new nodes, since limited observations are available about them, which results in an unreliable trust metric during those initial interactions with that node. An attacker may take advantage of this by continually joining and leaving a network, as is the case in the whitewashing attack discussed in Chapter 3. Therefore, there is a crucial need for a mechanism by which an observer, or wider network, can gain an understanding of a node's behaviour in the early stages of its participation in the network.

We propose a hidden Markov model (HMM) of a malicious node and show that it enables earlier classification of its behaviour than trust inference alone, thereby providing a means of protection against a node's attacks during that initial vulnerable phase. Additionally, the model is designed to separately take into account the malicious state of the network and the probability of attack. In conjunction, we show that Baum Welch estimation of the node's transition probabilities results in time sequential data that can be leveraged using a Long Short Term Memory recurrent neural network in order to classify the node earlier by predicting its future state and emission transition probabilities.

5.1 Introduction

Trust inference has been proposed as a possible security framework for distributed networks as it allows nodes to independently and accurately evaluate each other [141]. This provides a means of protection for low-capability devices in networks, by introducing accountability to node . However, existing representations of trust in the literature tend to treat the node's behaviour as a simplistic parameter, with limited exploration of the relationship between the node's deliberate actions and the outcome of those actions. Some models assume that malicious intent will always lead to a negative outcome [142]. On the contrary, intent and outcome are not linearly linked. Data from an honest node may not always be correct, whilst data from a malicious node may not always be incorrect.

This is especially true when considering trust attacks (TAs). In these, the attackers rely on manipulating their own reputation in the network prior to committing the attack, because a typical trust network will penalise nodes with negative reputations. Therefore, for the attack to have a significant impact on the network, the attacker will need to cooperate with the network for periods of time in order to build up a good reputation.

Trust attacks and, more broadly, insider attacks are a relatively well-explored topic in literature. However, owing to the inconsistencies in understanding about the symptoms and extent of these trust attacks, existing research tends to address very specific, known attack models. Instead, in previous Chapter 3, we proposed a unified tier system for the classification of trust attacks, as shown in Table 3.1, which recognises the severity of an attack by assessing its dynamic relationship with the network's trust system. In our previous work, we proposed a protocol for the detection of and coexistence with Tier 1 nodes, followed by a contextual

categorisation method to classify and detect Tier 2 nodes in the absence of real-world training data.

This chapter seeks to build on the idea of trust inference by developing a generalised HMM of a node that encompasses the relationship between a node's intent and the interaction's outcome. We use this model to analyze node such as by predicting future characteristics of the node. With regards to Table 3.1, this work is broadly situated in Tier 2, but the protocols proposed herein are designed with a view to, in future, tackling Tier 3 attacks, with only minor changes required. The motivation behind this is that over 30% of successful attacks now consist of never-before-seen, or *zero-day*, attacks [143]. Therefore, there is a need for a way to continuously track the behaviour and attack probability for nodes, without relying on knowledge about specific attack models.

5.1.1 Problem and Contributions

In their simplest form, the Tier 2 and 3 attack strategies would consist of two modes, *attack* and *cooperate*, between which the node would transition. The timing and probability of these transitions are characteristics of the attack strategy. There already exists a very simplistic form of this binary strategy, called the ON-OFF attack [144]. The existing ON-OFF attack involves the attacker cooperating with the network for a certain amount of time or until a threshold has been met (the 'ON' phase), such as some minimum reputation metric, after which the attack is committed (the 'OFF' phase) [145]. Attackers can then seek to build up their reputation prior to an attack in order to maximize its effect on the network.

However, the ON-OFF attack has only been discussed in literature as a prescriptive strategy, where typically the ON occurs periodically with a fixed period, or the ON-to-OFF ratio is already known. [146] provides an example of both of these

assumptions. This approach makes the standard ON-OFF attack significantly less effective and also less realistic in implementation, with typical simulation results showing that systems are able to suppress the attack within the first few attack cycles.

Secondly, due to trust inference relying on the availability of sufficient observations about the target node, the network is particularly vulnerable to attack by newer nodes in the network. Existing literature presents a range of classification protocols, including some that we have utilised in previous work such as support vector machines. However, there is a need to classify nodes as potential risks as early as possible after they join the network.

In response to the discussed challenges, and in pursuit of a more realistic malicious attack model, we considered the following:

1. **Generalised model of a malicious node:** Propose a hidden Markov model (HMM) for a developed ON-OFF strategy of attack for a typical Tier 2-3 node, but which can be easily expanded to other malicious nodes;

2. **Transition probability estimation and classification:** Investigate the effectiveness of using the Baum-Welch (BW) algorithm to estimate the HMM parameters of the node based on observations, and to subsequently classify nodes by their level of risk to the network;

3. **Early classification and prediction:** Predict the future behaviour of the malicious node by using a long-short term memory (LSTM) recurrent neural network (RNN) as single-step and multi-step predictors.

5.1.2 Related Work and State-of-the-Art

In related work, [146] discusses the optimisation of the ON-OFF insider attack but assumes frequency and timing of monitoring the channel is known. Similarly, [110] includes some typical values for energy consumption in a WSN. The paper talks about modelling ON-OFF attack in order to generate a dataset, but doesn't disclose how the ON cycle is determined.

With regards to HMM modeling of malicious nodes, [147] uses an HMM for the prediction of false data injection in a network. It is assumed that nodes can exist in three hidden states: Malicious, Suspicious and Authentic. The observations then consist of malicious, suspicious or normal activity. However, it is unclear how the raw data was obtained for the attack strategy. [148] formulates an HMM for the spread of malware based on the susceptible-infected-susceptible (SIS) infection model and using random propagation, but does not evaluate the infection risk from any particular infected node, which would have been analogous to the trust inference approach.

The current state-of-the-art for behavioural prediction is the auto-regressive integrated moving average (ARIMA), which has been widely regarded as the benchmark comparison for LSTM-based approaches. [149] finds ARIMA to produce more accurate long-term predictions for stock valuation than LSTM. However, LSTM can more accurately pinpoint the timing of changes. In stock prediction and related applications, ARIMA may be preferable because they require the observation and prediction of broader upward and downward trends in the data. However, in terms of near-term behavioural prediction, we have chosen LSTM because closely tracking short-range changes in transition probability is important, especially if the model is to be extended for Tier 3 nodes in the future.

5.2 Hidden Markov Modelling of Malicious Nodes

Let there be a malicious node in a distributed communication network. There are two primary perspectives involved in the network: the malicious node's opinion about the rest of the network, and the rest of the network's opinion about the malicious node. During the initial interactions by a new node i in a network, we can approximate the rest of the network nodes' views about i as being approximately the same. Therefore, we can model the problem as a two-agent scenario:

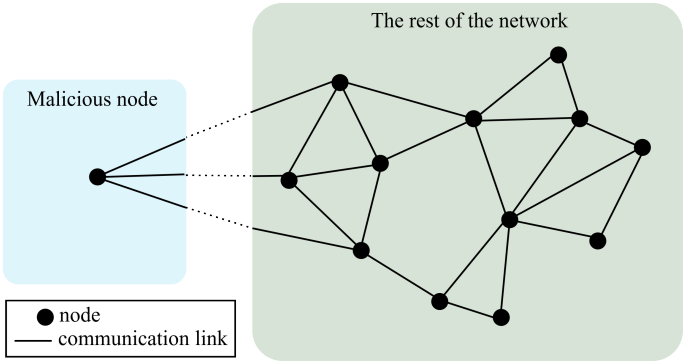


Figure 5.1: A two-participant approximation of the network.

Therefore, based on the assumption that the rest of the network disseminates information quickly between its nodes, we can approximate their collective views as a single entity as shown in Fig. 5.1. As a result, we propose a model for node behaviour from the perspective of *the rest of the network*.

5.2.1 Model Framework

A malicious node's goal is to disrupt the network by attacking it. An attack can take a number of different forms across various types of systems, as explored in our previous work.

Here, we define an *attack* as the intentional submission of incorrect data by the

source of that data. However, this does not guarantee that incorrect data will be received at the destination. Conversely, the receipt of incorrect data does not necessarily mean that the sender submitted incorrect data. From the network's perspective, only the received data is seen, and therefore this forms the emission states for our HMM in Fig. 5.2. The hidden states represent the current state of the node, and therefore the intent.

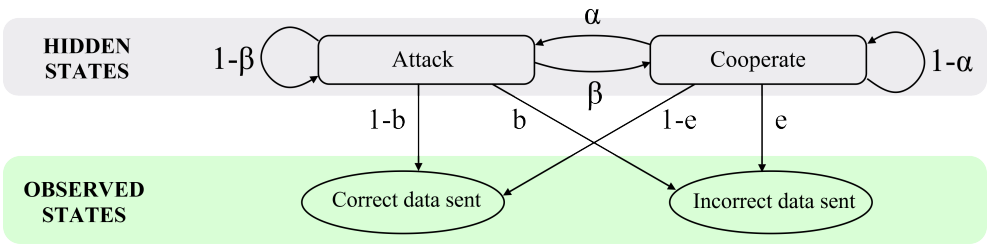


Figure 5.2: HMM from the perspective of the ON-OFF attacker.

We propose that the majority of node behaviour can be represented by this HMM, with varying values of transmission and emission probabilities. However, in the context of a potentially malicious node in a trust network, α and β are the node's control variables as determined by its strategy.

In many cases, the emission probability $b \approx 1$ since an attacker choosing to attack would almost always intend to send the wrong data. However, two factors may cause $b < 1$:

- Some models, such as [144], include a *mixed strategy* approach, where the act of attacking is considered a Bernoulli random variable even within the Attack state. This is to introduce further unpredictability in the malicious strategy.
- Secondly, an attack is not always successful. [145] discusses how a network's preexisting attack prevention strategy can be represented as a *mitigation coefficient* during each attack instance.

Initially, we assume $b \gg 1 - b$ meaning that when in attack mode, the attacker is highly likely to attack.

5.2.2 Model Challenges

There are three challenges to be aware of with regards to the adoption of the HMM approach.

1. **Need a pseudo-count for low-probability emissions:** When a particular hidden or emission state occurs with a very low frequency, it may not occur enough times in the training or test datasets for any estimation algorithm to converge to its accurate probability of occurrence. In the case of our proposed HMM, this makes it difficult to take into consideration the channel error without prior information about its probability. However, the channel error is something that can be estimated or measured using established techniques [150].

2. **Limited real-world training data:** Real-world training trust data is limited. The few applicable datasets, such as Advogato [151] and other *trustlet* data, tend to exhibit a low frequency of attack, leading to the zero-probability error. Some examples in the literature tackle this issue by setting a maximum number of iterations, or epochs, in training and estimation processes such that the learning algorithm does not fully converge to a zero probability for the low-frequency state. This may be suitable enough for broad classifiers such as image identification, but a significant drawback of this approach is that the overall accuracy of the system is reduced as a result of the limited training time. Another approach is to include a pseudocount of the low-frequency emission in the datasets, thereby setting a minimum value in the estimation of its probability.

3. **Need for initialisation:** The BW algorithm works by finding a local optima for the probability of the parameter set. Any BW solution to a particular HMM would still provide an accurate representation of reliability, perhaps even an alternative measure of trust, but only one solution would give an accurate representations of the variables α, β, e b. If the BW converges to a different local optima than expected, then the output may reflect incorrect values for these probabilities. To alleviate the risk of this, the BW initial estimates α, β and b, are set to reflect likely values of those probabilities, e.g., $\alpha_0 = 0.1, \beta = 0.8, e = 0.05, b = 0.9$ to reflect that $\alpha \ll \beta$ and $e \ll b$.

5.2.3 Relationship between τ and the HMM

Recall that the probability density function of trust by a node i about another node j is given as follows in the beta reputation model [33],

$$f(p_{i,j} \mid \omega, \psi) = \frac{\Gamma(\omega + \psi)}{\Gamma(\omega)\Gamma(\psi)} p^{\omega-1}(1-p)^{\psi-1}, \tag{5.1}$$

where Γ is the gamma function, $\omega = r + 1$, $\psi = w + 1$, and r and w are respectively the positive and negative observations about the target j. Thus, the expected trustworthiness is given by

$$\hat{\tau} = E(\tau) = \frac{\omega}{\omega + \psi}. \tag{5.2}$$

In Chapter 4, we defined effective trust as $p = \mathbb{P}(Y = X)$, and therefore

$$p = \tau(1 - e) + e(1 - \tau). \tag{5.3}$$

and, like τ, has a beta distribution, meaning that we cannot know exactly the value

of p but can estimate it by using observations,

$$\hat{p} = \frac{r+1}{r+w+2}. \tag{5.4}$$

Letting $p = \mathbb{P}(Y = X)$ as above, it follows that $\hat{p} = E(p) = \mathbb{P}$(correct data sent), with reference to the proposed HMM in Fig. 5.2. Therefore, the relationship between $1 - \hat{p}$ and the HMM attack transition probability α is dependent on the current hidden state as follows:

$$\mathbb{P}(\text{incorrect data sent}|\text{attack state}) = (1-\beta)b + \beta e, \tag{5.5}$$

$$\mathbb{P}(\text{incorrect data sent}|\text{cooperate state}) = (1-\alpha)e + \alpha b. \tag{5.6}$$

Note that in contrast with $1 - \hat{p}$, α additionally takes into account the node's probability of sending incorrect data separately from the probability of being in the attack state. The attack probability b is set close to 1 in this chapter, but the model has been designed such that in future, the HMM dynamics can be explored for $b < 1$ in order to reflect Tier 3 malicious nodes from Table 3.1.

5.2.4 Training Data, Definitions and Initial Simulations

In this section, we present the preliminary results from running a BW estimator on our HMM in Fig. 5.2. These will then form the basis for the early classification work in later sections.

As discussed in Section 5.2.2, there is extremely limited real-world training data for trust networks. Therefore, we generate training data by passing generated or chosen values of α, β, e and b into Matlab's *hmmgenerate* function. This function outputs observation sequences of the desired length based on the chosen

parameters.

The resulting dataset is then analysed using a BW estimator that has no prior knowledge about α, β and b. In other words, the values chosen during the training data generation are not known to the BW estimator, nor to the LSTM classifier that is subsequently used to analyse the BW estimates.

We measure time in *rounds*, where each round represents one observed interaction with the node. Depending on the rate of interactions per second for a particular network, it would be fairly straightforward to convert between seconds and rounds.

The first result compares the accuracy of effective $1 - \hat{p}$ with the HMM approach α, which are both representations of the node's *lack* of trustworthiness and are both calculated based on the same set of generated observations. $1 - \hat{p}$ is calculated based on equation (5.4), whilst the estimate for α is calculated using BW estimation algorithm. Both $(1 - \hat{p})$ and α were simulated over identical sets of training data.

Note that we define *convergence*, or *convergence time*, as the number of rounds after which the estimate converges to within 5% of their final estimates.

Secondly, we plot the convergence rate for each BW estimated parameter with relation to the true values of α and β. And finally, the BW estimates are reevaluated at each round. By treating the rounds as time steps, we can generate time sequence data for the estimates. Each iteration of the BW can be output as a single data sequence, or *BW trace*.

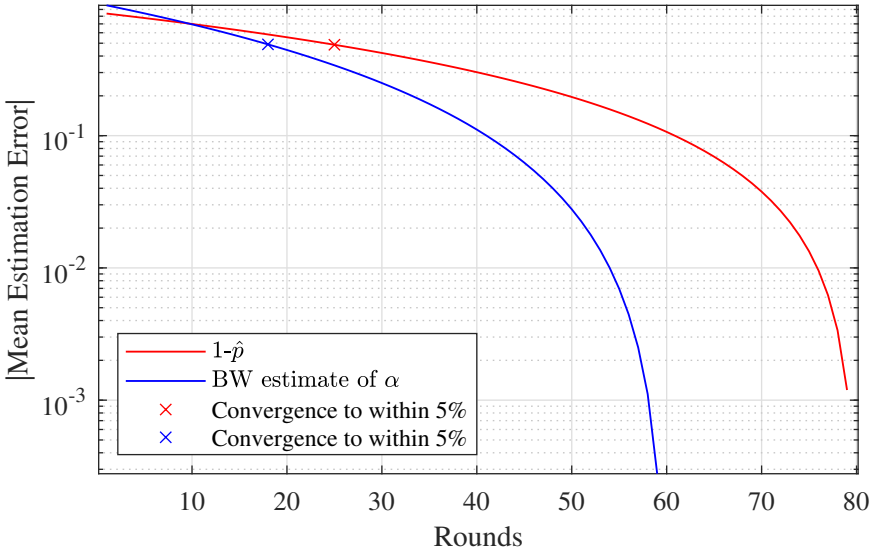


Figure 5.3: Error when estimating the maliciousness of a node using the proposed BW approach vs. statistical trust inference, for randomised values of α, and where $\beta = 0.6, e = 0.05, b = 0.95$.

5.2.5 Preliminary Results using Baum-Welch Estimator

Initial results in Fig. 5.3 showed that the BW estimator was able to arrive at accurate estimates for the transition and emission probabilities within the first 50-60 rounds. Therefore, in subsequent simulations, we will specifically seek to observe the first 50 rounds of interaction with the node.

The results in Fig. 5.3 shows that the BW estimator converges 7 rounds before $\hat{\tau}$, and also exhibits a lower estimation error from 10 rounds onwards. The convergence of α is dependent on not only the training data but also by the convergence of the other HMM parameters. In other words, if β takes more rounds to converge, then so too would α, since their estimates are interdependent.

To illustrate this dependance, we simulated the BW estimator and observed how the convergence times for α and β are affected by their actual values. Figs. 5.4

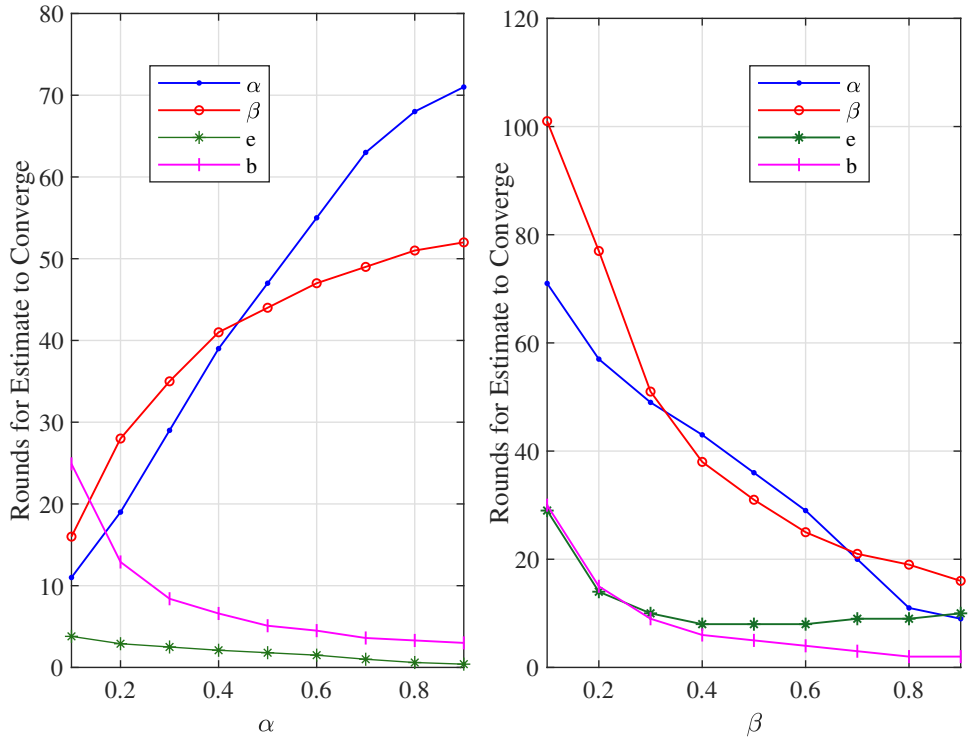


Figure 5.4: Effect of α and β on the number of rounds taken by the BW estimator to consistently achieve within 5% of the true transition and emission probability. With $\beta = 0.5$ in the first (left) result, $\alpha = 0.5$ in the second (right) result, and $b = 0.99, e = 0.05$ in both.

shows the results of this. Given that $\alpha_0 \ll \beta_0$, it follows that uncharacteristically high values of α and low values of β result in high convergence times. This is an unlikely scenario, but were it to occur, a malicious node could take advantage of the high convergence time by attacking frequently as soon as it joins the network.

Therefore, the convergence time should be minimised as much as possible. Interestingly, the BW estimate of an HMM parameter looks very similar in every iteration. For example, with $\alpha_0 = 0, \alpha = 0.3$, the BW estimate over any number of rounds would follow a similar trajectory every time.

This can be illustrated by plotting the BW estimate iterations, or traces, side by

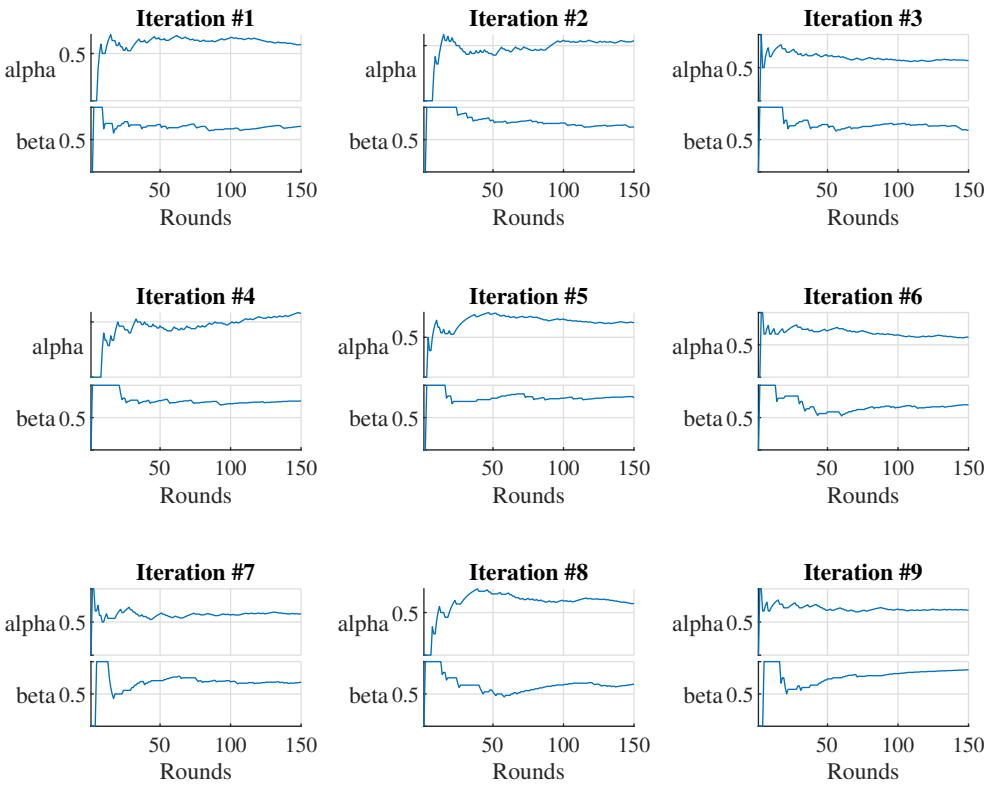


Figure 5.5: BW estimations of the HMM transition probabilities when $\beta = 0.6, \alpha = 0.3, b = 0.99, e = 0.05$. Each subplot represents an independent run on the same sequence of observations.

side. Fig. 5.5 shows the results when the BW estimator is run on the same set of training data nine times, and highlights the similarity in trajectory. Based on this, we propose in the next section that the BW results form a set of time sequences, which can be used to improve estimator accuracy in the first few rounds of a node's interactions.

5.3 Time Sequence Analysis Using an LSTM

The preliminary BW results have shown that the estimation arrives at an accurate values for the transition and emission probabilities of the HMM after the initial 50-60 rounds. It is during those initial rounds that the network is most susceptible

to an attack. Recall that each round is an interaction with the target node. Depending on the rest of the network's frequency of interactions with that node, it can take a significant period of time for which there is not sufficient observations about the node to accurately identify it as malicious, if it is. This creates a period of vulnerability for the network every time a new node joins. In this section, we will examine the possibility of using an LSTM RNN in conjunction with the BW estimates over time, with the aim of deducing the round 50 (stabilised) BW estimates much earlier than round 50. In addition to early classification, we explore the possibility of using our proposed BW-LSTM method for extrapolating node behaviour and predicting the next-step and near-future α values of a node.

5.3.1 Proposed System Layout

The overall BW-LSTM system layout is given in Fig. 5.6. So far, we have discussed the BW estimation. The output from the BW training data is passed through class assignment, exponential smoothing and then into the LSTM. The LSTM is then used to (i) classify nodes as per their level of risk to the network, (ii) predict their α for the next interaction, and (iii) predict their α multiple rounds into the future.

5.3.2 LSTM Classification based on α: Sequence-to-Class

In order to understand the behaviour of a node in a network as soon as possible after it joins the network, its initial behaviour needs to be extrapolated such that its eventual Attack transition probability is known.

LSTM classification reads labelled training data as input and learns the patterns of behaviour therein via regression. Then, when presented with new data, it can classify the sequence based on this knowledge. However, the BW estimator does

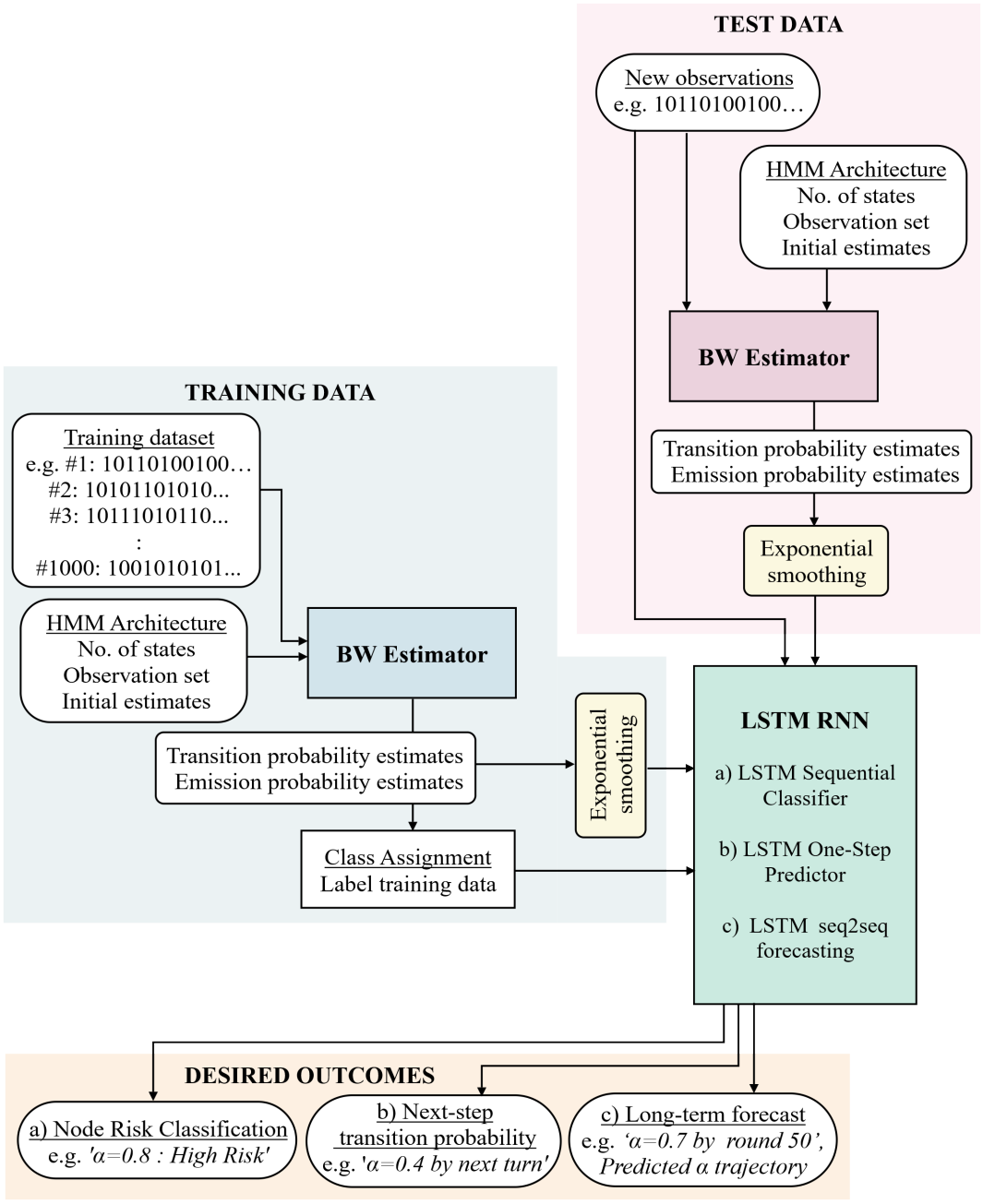


Figure 5.6: Block diagram of the BW-LSTM sequential classifier.

not inherently classify or label its output, and so we assign the training data *classes* based on the nodes' Attack transition probabilities α.

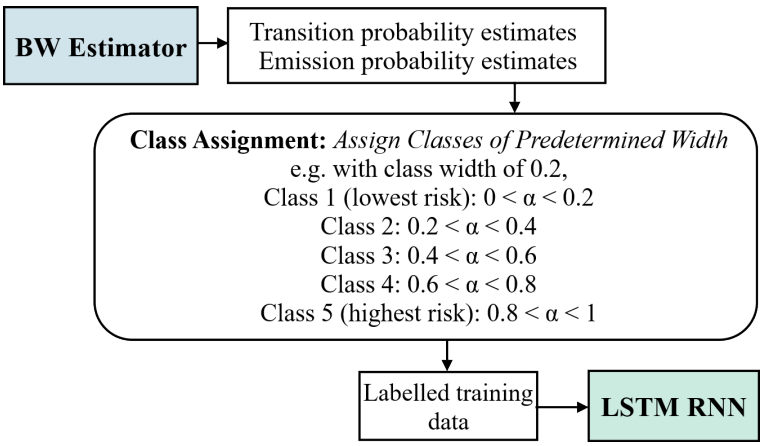


Figure 5.7: Block diagram showing the interface between the BW Estimator and LSTM RNN. The class assignment results in a labelled training dataset, which becomes one of the LSTM inputs.

The class assignment function divides $\alpha \in 0, 1$ into classes of fixed width. Our default class width is 0.2, resulting in five different classes. As shown in Fig. 5.7, Class 5 represents the highest risk of the node transitioning into the Attack state. In terms of implementation, the system response could be tailored to each class rather than a blanket response for "honest" nodes or "unreliable" nodes. Having access to classes provides a system with the opportunity to utilise its knowledge about a node, as we have explored in Chapter 4, where we used input from highly unreliable nodes to reinforce a data aggregation output.

Let the LSTM classification accuracy be defined as follows,

$$\text{Classification Accuracy} = \frac{\text{No. of correctly classified data series}}{\text{Total data series}}. \tag{5.7}$$

The choice of class width presents a trade-off. The narrower the class width, the more likely the LSTM classifier will be to mis-classify a node. However, the incorrectly predicted class is highly likely to be adjacent to the correct class. In this case, narrower classes may actually perform better overall, since the difference in

α between the LSTM estimated class and the true class is smaller. Therefore, we define a second performance metric,

$$\text{Mid-point error} = |\text{Mid-point of estimated class} - \text{mid-point of true class}|, \tag{5.8}$$

which allows us to observe the overall accuracy of the LSTM, even when the estimated class is not correct.

5.3.3 Pre-Processing LSTM Input Using Exponential Smoothing

The BW estimate for Attack transition probability α converged by about round $r = 50$, but we propose that an LSTM can be used to predict α at $r = 50$ via sequence-to-sequence prediction. However, as this work focuses on the initial 50 rounds of interactions with the node, each new interaction with the node has a non-negligible effect on the BW estimates of the HMM parameters. For example, although the overall trends are clear in Fig. 5.5, there is visible jitter in the BW traces. Pre-processing the data before running the LSTM has been shown to improve the performance of the LSTM classifier and predictors [152].

Therefore, we use smoothing to remove the more high frequency components of the BW time series trace, which will allow the LSTM to focus on the underlying trends, and a comparison of two smoothing techniques is provided in Section 5.4.2

5.3.4 Next-Step Prediction using LSTM: Sequence-to-Prediction

The LSTM single-step predictor is trained like the LSTM classifier, but instead of classifying the sequence, it outputs the most likely next-round values of the desired parameter. The main difference between LSTM prediction and classification

is in its implementation.

Predictions about future data can be made by time-shifting the labels in the training data, such that label for a training data series is mapped to the value of that data series Δr rounds in the future. An example with $\Delta r = 2$ is shown in Fig. 5.8. Note that for single-step prediction, $\Delta r = 1$.

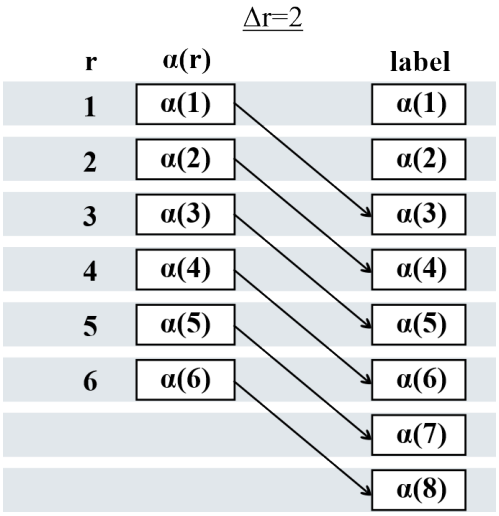


Figure 5.8: Illustration of time-shifting for LSTM prediction, showing a time shift of $\Delta r = 2$ rounds.

5.3.5 Multi-Step Forecasting using LSTM: Sequence-to-Sequence

To achieve early classification of a node, its α needs to be accurately estimated much before a trust inference algorithm or BW alone would be able to. The preliminary BW results showed that the BW estimate of α converged by round $r = 50$. By utilising an LSTM sequence-to-sequence estimator, it would be possible to estimate α in advance by Δr rounds. Then, for example, the converged value of α at $r = 50$ can be predicted by applying a time shift of $\Delta r = 20$ to the data series from $r = \{1, ..., 30\}$, or even earlier by applying $\Delta r = 25$ to the data series from $r = \{1, ..., 25\}$.

Table 5.1: Default parameter values for simulations.

Parameter	Default value
α	0.3
β	0.75
e	0.05
b	0.95
No. training data series	1000

Parameter	Default value
α_0	0
β_0	1
e_0	0.05
b_0	1
LSTM class width	0.2

The sequence-to-sequence estimator produces predictions for future values of α using the time-shifting method described in Section 5.3.4, but with $\Delta r > 1$.

5.4 Results & Discussion

5.4.1 Method

The LSTM is structured as three gates: the forget gate, the input gate, and the output gate. The forget gate determines how much of the previous cell state to retain, the input gate determines how much the input should affect the cell state and the output gate determines how much of the updated cell state to commit into long-term memory. A simple 4-layer LSTM RNN was used for the simulations:

Algorithm 2 The LSTM RNN set-up in MATLAB

```
layers =
[sequenceInputLayer(numChannels)
lstmLayer(128)
fullyConnectedLayer(numChannels)
regressionLayer] =0
```

Unless otherwise stated, the true values and initial estimates are those listed in Table 5.1.

5.4.2 Data Pre-Processing Results

The cross-correlation between BW traces is adversely affected by small-scale fluctuations, or jitter, in the time series data, caused by iterative over- and under-estimation by the BW. In order to reduce this jitter and extract only the trends required for the LSTM analysis, we apply a smoothing function. Fig. 5.9 presents the simulation results from a direct comparison of moving average smoothing versus exponential smoothing, the functions for which are given in Appendix D.

Consistently, the exponential smoothing is more sensitive to large gradient changes in the data. This can be seen because the exponentially smoothed curve's maxima and minima occur sooner after the respective maxima and minima in the raw (unsmoothed) data. Given that the aim is to identify malicious behaviour as soon as possible, exponential smoothing is more suitable. Secondly, the moving average smoothing suffers from lower accuracy where the data experiences large sudden changes, as can be seen by the false peaks in the blue trace whenever the unsmoothed data troughs. The window used in the moving average can be reduced to counter this problem, but that limits the smoothness possible using the moving average method.

Note that Fig. 5.9 is a single-shot example for illustration purposes, but further simulations were done to confirm that these results are consistent. Based on this, we chose to use exponential smoothing.

In order to optimise the performance of the BW-LSTM classifier and predictors, it is important that the BW traces have a strong cross-correlation with each other for the same configuration of parameters. The cross-correlation of the traces are affected by two factors, the initial estimates for each parameter and the pre-processing of data via smoothing. Fig. 5.10 shows the cross-correlation between

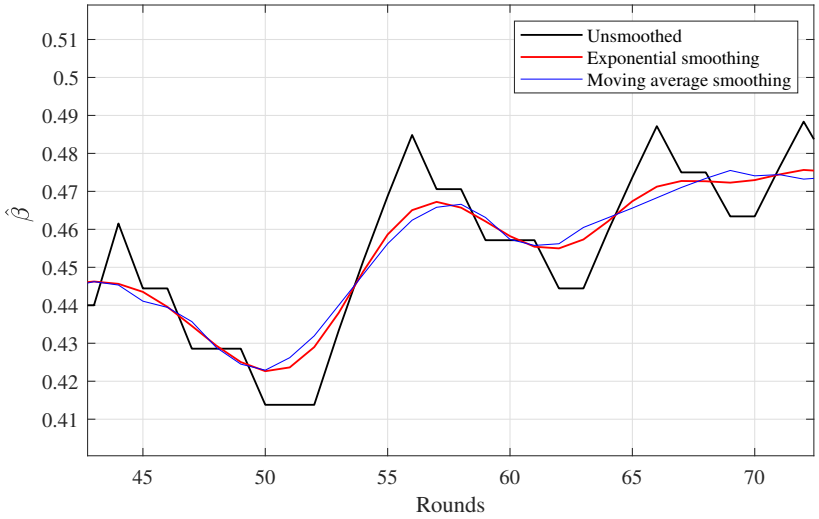


Figure 5.9: Comparison of exponential and moving mean smoothing techniques with smoothing factor $\epsilon = 15$ and a window size of 5 respectively. This is a single-shot example to illustrate the difference in behaviour between the two smoothing functions on a sample segment of a $\hat{b}$ BW trace.

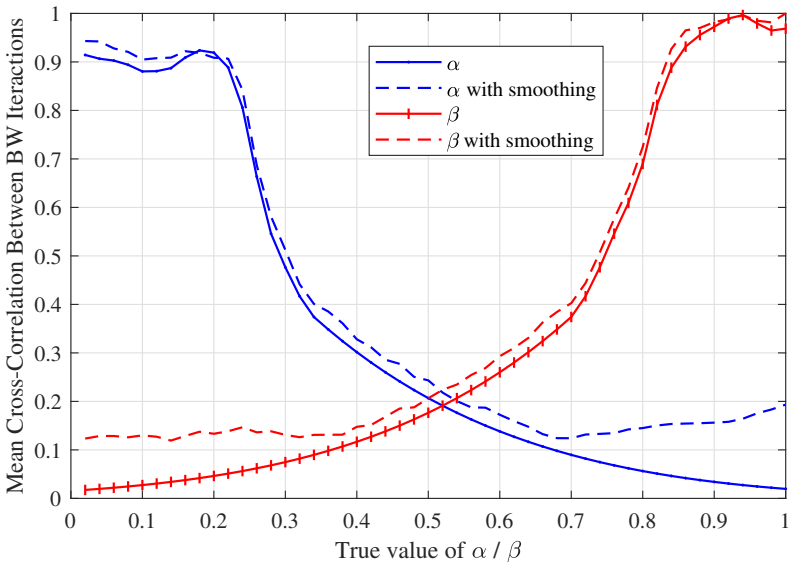


Figure 5.10: Mean cross-correlation across BW traces over the first 50 rounds, where the initialised BW values are $\alpha_0 = 0.2, \beta_0 = 0.9$.

the BW traces with $\alpha_0 = 0.2, \beta_0 = 0.9$, and smoothing factor $\epsilon = 15$ (refer to

Appendix D for full equation).

The results show that local maxima occur in the cross-correlation at the initial estimate values, which is to be expected because this causes the trajectory of the BW α estimate to be much flatter. In other words, the initial estimate already fulfils the requirements of the BW and so the trace becomes more predictable. Secondly, the exponential smoothing provides a consistent improvement in the

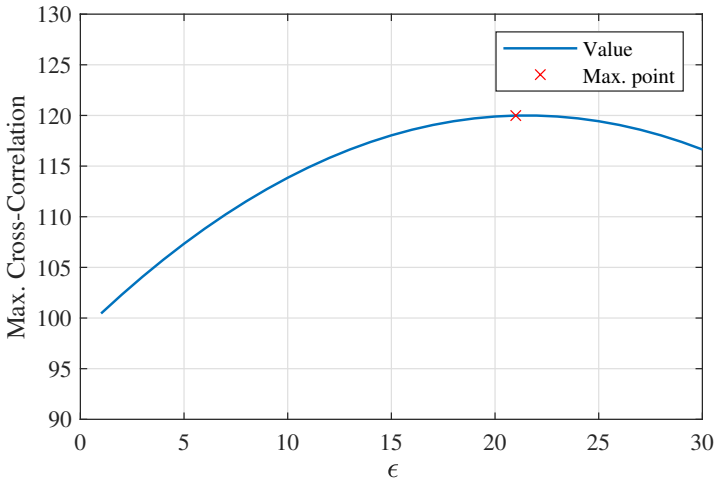


Figure 5.11: Maximum cross-correlation across BW traces over the first 50 rounds, where the HMM transition probabilities are $\alpha = 0.3$ and $\beta = 0.7$. Optimal smoothing factor is $\epsilon = 21.5$.

cross-correlation between traces. The value of ϵ needs to be balanced such that it is high enough to improve the cross-correlation whilst not being so high as to decrease the uniqueness of the trace for that particular configuration of parameters, i.e., *under-fitting*.

Therefore, to decide an optimal value for ϵ, we applied exponential smoothing with different values of ϵ to the BW traces, and the resulting cross-correlations are plotted in Fig. 5.11. From this, it can be deduced that $\epsilon = 21.5$ is a suitable smoothing factor for pre-processing the data.

5.4.3 LSTM Classification Results

Figure 5.12: Visualization of the LSTM training procedure, run in small batches of 27 data samples for 2700 total datapoints.

Figure 5.13: Confusion matrix of the classifier output vs. the labelled test data. High risk defined as $\alpha > \frac{2}{3}$, low risk is defined as $\alpha < \frac{1}{3}$.

Fig. 5.12 shows the training process for the BW-LSTM classifier. It is promising to see that the RNN is able to achieve around $80 - 90\%$ classification accuracy within a relatively small number of training iterations. Initially, the BW-LSTM was run as a broad classifier with class width of $0.3\dot{3}$, such that $0 < \alpha < \frac{1}{3}$ corresponds to Low Risk, $\frac{1}{3} < \alpha < \frac{2}{3}$ corresponds to Mid Risk, $\frac{2}{3} < \alpha < 1$

corresponds to High Risk. The results are shown in Fig. 5.13. It can be seen that the most common form of misclassification occurs where 20 High Risk nodes were falsely classed as Mid Risk. However, the majority of nodes have been correctly classified.

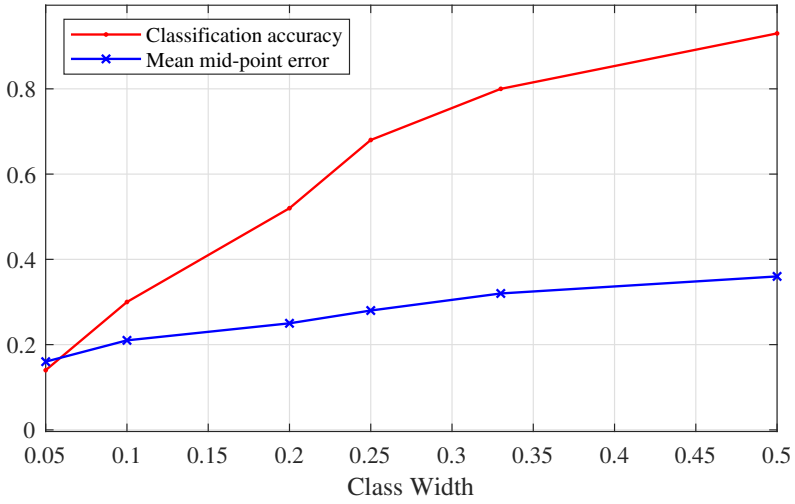


Figure 5.14: Effect of class width on classification accuracy and mid-point error.

As expected, the class width is shown in Fig. 5.14 to present a trade-off between minimising mid-point error and maximising the classification accuracy. As the class width increases, so too does the classification accuracy as a result of the more pronounced difference in the BW traces between two adjacent classes. Similarly, with a smaller class width, there is a higher probability that neighbouring classes will have comparable BW estimation traces, thereby resulting in a higher probability of false classification. It follows naturally from this that the mid-point error, as described in (5.8), is lower for smaller class width, where there is a smaller difference between the α mid-point of each class. The implication of this is that, where it is more important to accurately find α rather than to correctly classify the node's risk level, it is preferable to choose a smaller class width despite the lower classification accuracy.

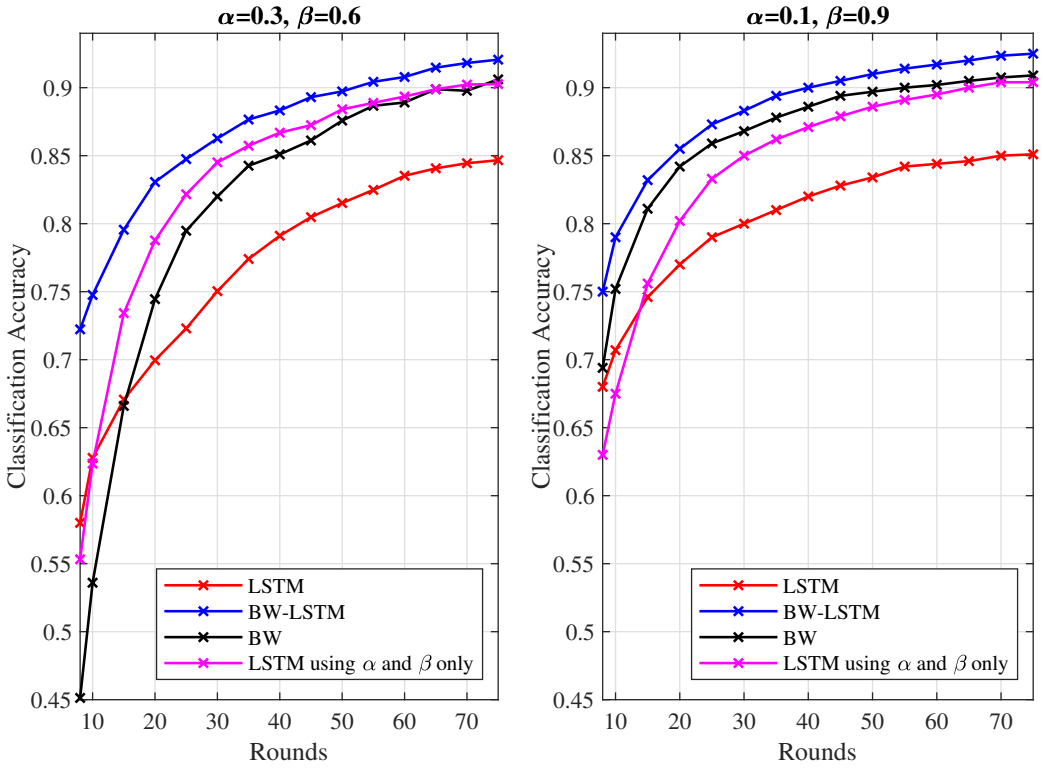


Figure 5.15: Comparison between BW-LSTM and component methods when classifying nodes after each round of data, where $b = 0.99, e = 0.05$.

A comparison of four methods is shown in Fig. 5.15. LSTM and BW are separately simulated, with the latter exhibiting a significantly higher classification accuracy. LSTM using α and β data only is similar to BW-LSTM in that it utilises the BW estimates for these two parameters, but does not have access to the raw data. Its performance is comparable to that of the BW estimator on its own. Our proposed method, BW-LSTM, exhibits a consistently higher classification accuracy. Importantly, there is a notable improvement in the BW-LSTM accuracy for less than 30 rounds, which was the hoped outcome and the key motivation for applying the LSTM.

As expected, the more extreme (closer to 0 or 1) values of α and β result in a

significantly faster convergence for all of the compared methods. This is because values closer to 0 and 1 result in lower uncertainty, but have a minimal effect on the final classification accuracy after 60-70 rounds. Notably, the BW accuracy is most improved in the second graph, and this subsequently results in the improvement of the BW-LSTM classification accuracy, though the margin between the two is significantly reduced. However, regardless, in the lower range of rounds, BW-LSTM retains a significant margin in terms of its accuracy when compared to the alternatives.

5.4.4 LSTM Prediction Results

The results of the single-step BW-LSTM predictor are given in Fig. 5.16, showing comparable performance with the current state-of-the-art sequential prediction algorithm ARIMA. The LSTM predictor exhibits a tendency to over-estimate, leading to a higher overall next-step prediction accuracy in the ARIMA algorithm. However, the ARIMA shows a more pronounced lag in its predictions, while the BW-LSTM approach follows sudden changes in the BW trace more closely.

Importantly, the multi-step prediction results in Fig. 5.17 shows that the lag in the ARIMA becomes more pronounced for long-term prediction. The higher accuracy in the BW-LSTM prediction is likely due to it being trained on many similar BW traces. Therefore, it is able to effectively remember where severe fluctuations in the BW trace from similar cases in the training data. In contrast, being based on a moving average approach, the ARIMA algorithm is adversely affected by high, unexpected gradients in the data series. This can be seen in its multi-step prediction of α, where the positive gradient just before the end of the smoothed data has caused a significant over-estimation of subsequent values.

We simulated both BW-LSTM and ARIMA over a range of time shifts for the

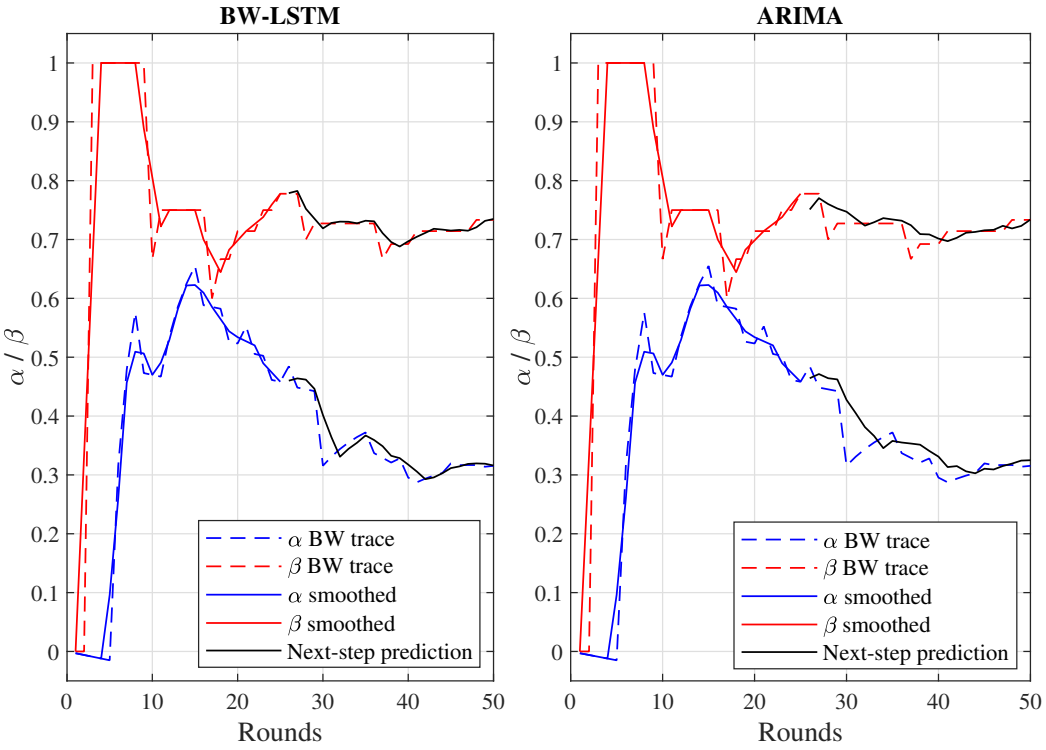


Figure 5.16: Single-step BW-LSTM prediction, $\Delta r = 1$, a comparison between the proposed BW-LSTM and current state-of-the-art approach ARIMA.

same set of training data. The two methods were used to estimate α by Δr rounds in advance. BW-LSTM is able to more accurately predict the value of α for $\Delta r > 10$. This is in line with previous results, but confirms that our proposed BW-LSTM is the superior choice for predicting the state transition probabilities of our HMM representation of a network node.

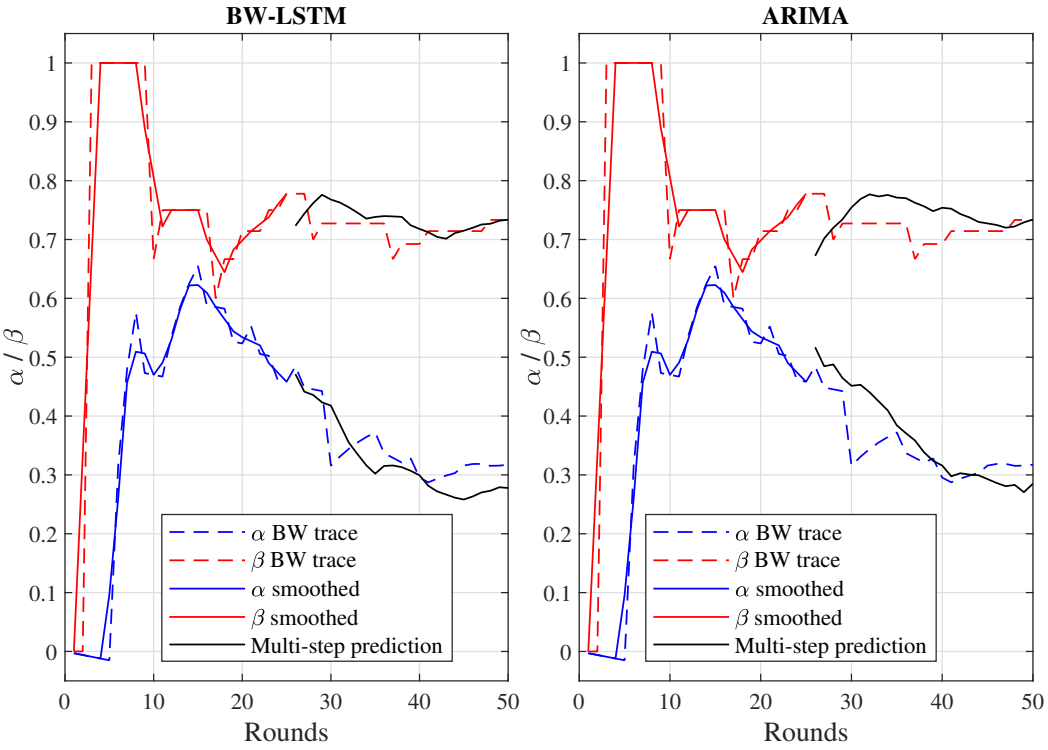


Figure 5.17: Multi-step BW-LSTM prediction, $\Delta r = 25$, a comparison between the proposed BW-LSTM and current state-of-the-art approach ARIMA.

5.5 Summary

5.5.1 Discussion on Implementation

In previous work, we proposed a trust-based data aggregation protocol that utilises knowledge about a node's trustworthiness to reinforce the aggregate result, regardless of whether that trustworthiness is high or low. However, the work concluded that the utilisation of trust alone left the network vulnerable in the early stages of a node's participation, before enough observational evidence can be gathered about it such that its reputation value is accurate.

Instead, it would be particularly advantageous to understand and preempt node

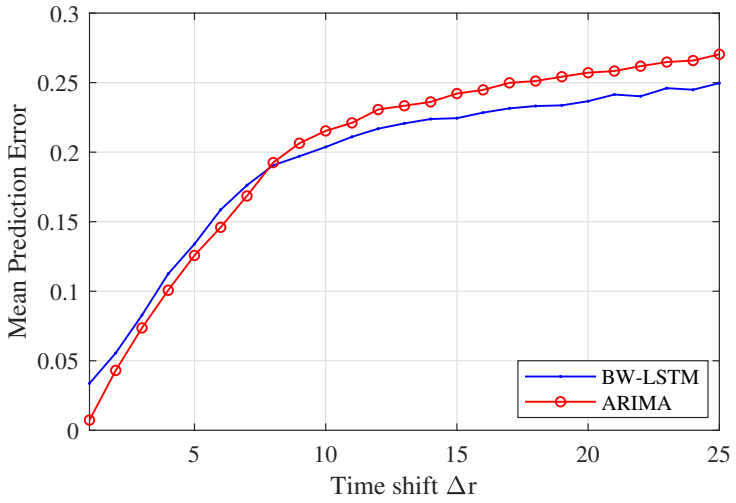


Figure 5.18: Mean prediction error vs. Δr, where the α prediction error is measured at the 50^{th} round each time. For example, where Δr on the x-axis is 10, the data series used to inform the prediction would have included the first 40 values from the BW trace.

behaviour from a security standpoint. Knowing that a node's attack transition probability α is likely to converge to 0.9, for example, would allow a network to decide to treat data from that node with caution, or otherwise utilise the knowledge that a node is potentially a threat such as our trust-based aggregation protocol.

Additionally, being able to classify nodes early presents an opportunity to delegate system responsibilities according to a node's characteristics. For example, a node experiencing high channel error e, perhaps due to its lack of proximity from the rest of the network, would not be a suitable choice to whom to delegate time-sensitive network tasks. Similarly, a node that is predicted to have $\alpha < 0.3$ would be a suitable choice for relaying messages as the node would be less likely to intentionally alter the data, and thus this could be used to inform a routing algorithm.

The proposed HMM framework could easily be adapted to reflect a broad range of network attacks. For example, in a malware attack, the attack transition probabil-

ity $\{\alpha, \beta, (1-b), e\}$ translate readily as $\mathbb{P}$(infection), $\mathbb{P}$(cure), $\mathbb{P}$(resisting infection), and $\mathbb{P}$(channel error) respectively, and the same format of data can be used to track the infection probability over time across the network.

The HMM can also be adapted in order to adjust the trade-off between processing complexity and how early a classification can be made. Much like in Chapter 3, the specific set-up of the ML/AI tool being used can heavily influence the overhead placed on the network by our approach. Unlike at the beginning of our research five years ago, these tools, such as a basic LSTM framework, are now readily available and easily adaptable through pre-defined functions in Matlab, Visual Studio, etc. This makes it all the more accessible to integrate our proposed approach into a wide range of distributed communication networks.

5.5.2 Conclusion

In this chapter, we have presented an HMM representation of a network node which, unlike existing trust inference frameworks, reflects not only the intent of a network nodes but also the relationship between this and the outcome of interactions. Based on the proposed HMM, a BW estimator was used to test data in order to estimate the transition and emission probabilities of the node HMM with a high accuracy. The BW on its own took around 50 interactions to converge to correct estimates, but with the use of an LSTM RNN, it was shown that accurate estimations could be produced at even round 25 with the use of the BW-LSTM multi-step predictor. The performance of the proposed BW-LSTM approach was better than comparable algorithm ARIMA for preempting the value of α by up to 25 rounds.

The proposed HMM provides a foundation for expanding the early classification to Tier 3 nodes, as per Table 3.1. This is out of the scope of this chapter, but it

would be beneficial to be able to generate estimated projections of a node's future behaviour, not just during its initial phases but continuously during its time in the network.

This would require an adaption of the HMM to include dynamic probabilities. Furthermore, for expansion of the approach to prevent Tier 3 attacks, it would be necessary to train the LSTM based on the BW traces from Tier 3 malicious nodes. Despite the lack of real-world training data, [154] suggests that there is consistently a correlation between the behaviour preceding similar attack strategies, and it would be expected that this would be true of Tier 3 attacks too. Therefore, expansion of the HMM framework and BW-LSTM approach form part of our future work [153]. Finally, it would be necessary to adapt the HMM model parameters based on real-world data, for which distributed communication network data will first need to become more widely available.

6 | Conclusion and Future Work

Over the last fifty years, the number of wireless communication devices has rapidly grown. Currently, it is inevitable that there needs to be a shift towards, or some incorporation of, distributed network architectures in order to alleviate the load on centralised network hubs, such as cellular base stations. Furthermore, the consumer expectation of service quality and speed has driven the interest into futuristic networks, such as IoT and WSNs.

Trust inference will play a significant role in providing distributed security to these networks, particularly where highly demanding cryptographic security is not within the scope of the nodes' abilities. However, despite its suitability as a distributed network security framework, trust inference itself has vulnerabilities that need to be addressed before implementation. Although the literature has begun to address some of those vulnerabilities, existing solutions either take on a targeted approach that only defends against a subset of attacks, or makes very specific assumptions about the intentions of the malicious node.

Therefore, there is a need for a more comprehensive view of threats to a trust network, a plausible protocol for the integration of trust data into distributed network functionality, and a global model for malicious node behaviour that readily facilitates analysis and extrapolation of behavioural trends. This thesis has explored these aspects across the three research chapters.

In Chapter 3 we presented a unified framework of trust network threats, designed specifically from the perspective of a security protocol. This was done by comparing and collating trust attack strategies from across the literature based on attack symptoms and requirements, as would be seen by an observer. This framework then formed the basis for a novel contextual characterisation method for classi-

fying malicious network nodes, which provided a means of identifying potential attacks before they happen. By plotting the network nodes within the proposed parametric space, a support vector machine was successful in identifying possible malicious nodes and classifying them based on their attack strategy. Additionally, our proposed method was able to classify malicious nodes and their strategies even when the parametric space was flooded with honest nodes, and provided significantly better performance than the closest equivalent unsupervised approach. Having shown that it is possible to identify and classify malicious nodes, it followed that the next step would be to understand how knowledge about a malicious node can be used to reinforce distributed network security and functionality.

Based on this, in Chapter 4 we designed a data management and aggregation protocol for a fully distributed, trust-based network. The proposed data management algorithm utilised elements of blockchain technology in order to distribute the storage of any given piece of information across the network. The overhead of this on the network nodes was significantly reduced through the design of an accompanying trust-based aggregation protocol, which was able to not only aggregate data from multiple nodes with a very high accuracy, but also utilised trust information about those nodes to reinforce the aggregate result, regardless of whether the nodes were trustworthy or not. Furthermore, it was shown that the distributed data management and aggregation protocols could be easily applied to packet-based data with the use of LDPC error correction. This work also expanded on the idea of trust probability by accounting for intent and channel error separately.

We then seeked to expand on the idea of intent in Chapter 5. As already discussed in Chapter 3, nodes can have varying levels of awareness about the network's trust protocol. With reference to Table 3.1, we designed in this chapter a comprehensive HMM for malicious nodes, which can account for the nodes' intent, channel error, and mixed strategy attack protocols. By applying a BW estimation algorithm, it

was found that the estimate over time could be plotted as a temporal graph, which could then be analysed by an LSTM RNN in order to classify and extrapolate the node's behaviour. It was shown that the proposed BW-LSTM method was able to successfully classify a node's attack transition probability as early as 20 interactions after it first joined the network. Additionally, the BW-LSTM method presented a higher accuracy than the current state-of-the-art approach ARIMA.

In terms of implementation, there is significant scope for the direct application of this work into real-world systems. Our proposed protocols across this thesis can be implemented in conjunction, as shown in Fig. 6.1, to provide security to a network throughout its lifetime. Although distributed communication networks are not yet widely in use, our proposed protocols can be applied to existing trust environments, such as social media, online retail, and peer-to-peer services. In particular, the distributed data management and aggregation protocol can readily be applied within packet-based data transmissions, or within early distributed networks such as drone WSNs [25]. These protocols have the potential to provide a significant level of secure self-management for WSNs, given that they were specifically designed to provide independence to partially disconnected network clusters, and drone networks often suffer from partial disconnection due to their deployment in remote areas [49].

6.1 Future Work

A primary goal of any future work will be to address Tier 3 malicious nodes, as per the framework that we have proposed in Chapter 3.

The research towards achieving this goal can be divided into two categories. The first includes that which would directly continue the line of questioning of this thesis, whilst the second comprises the interesting but somewhat tangential questions

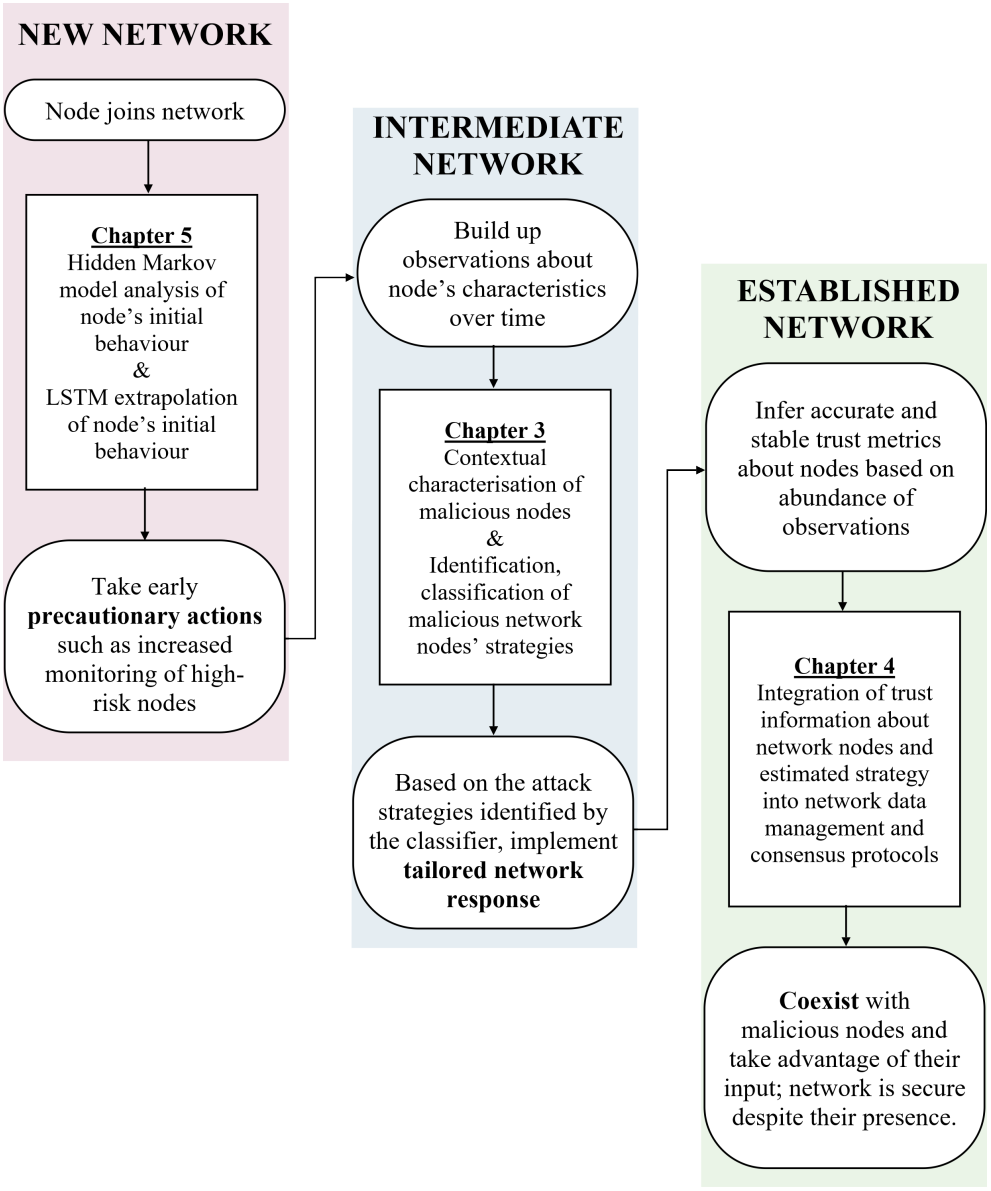


Figure 6.1: Possible implementation of the protocols proposed in this thesis in conjunction across the lifetime of a trust-based network.

that have emerged during the course of this research.

The first category would largely involve the continuation of our work on node behaviour extrapolation and prediction. This will be a required tool for the detection and prevention of Tier 3 malicious nodes with respect to our proposed attack tier

framework in Table 3.1. Naturally, the earlier a node's behaviour can be understood as malicious, the more likely it is for a network to be able to mitigate any attacks. This category includes:

- Chapter 3 presented a parametric representation of nodes within a distributed network. It may be possible to obtain real-world training data for Tier 3 attacks by plotting other, existing trust-dependent networks in a similar parametric space in order to observe behavioural patterns. Existing networks may include social media platforms, online retail marketplaces, or peer-to-peer services. [50] analyses the similarities between human trust networks and trust inference for network security.
- Chapter 4 specifically looked at the design of a fully distributed and secure data management protocol. Trust data was used to reinforce data using input from known malicious nodes. It would perhaps be possible to expand the aggregation protocol to also facilitate distributed resource allocation.
- In addition, it would be beneficial to design a formal test for the complexity of any proposed protocols. Although, where appropriate, we have measured this through the execution time of the algorithms that we have proposed, a formal complexity study would allow us to quantify the advantage of our IoT-accessible proposals versus the existing alternatives.
- Chapter 5 presented an HMM with the specific consideration of extending its analysis in future to Tier 3 malicious nodes. This can intuitively be explored by setting time-varying transition and emission probabilities within the HMM to reflect the changing strategy of the Tier 3 malicious node. The dynamics of these probabilities can be based on the contextual characterisation method for trust attacks, as presented in Chapter 3.

The second category of future work includes a range of possibilities that did not

fall into the direct trajectory of this project, but that would be interesting to explore in the future. Some examples are:

- With respect to the parametric space presented in Chapter 3, it may be possible to analyse the trajectories of nodes over time through the parametric space. We had investigated the possibility of using a convolutional neural network to extract features from those trajectories, which would allow for the mapping of particular *shapes* to known pre-attack strategies. However, the lack of real-world data upon which to train the neural network limited the scope of this investigation. Perhaps, once distributed network data is more widely available, this idea can be revisited. [51] has similarly extracted common features from malware code by applying a deep neural network on a visual representation of the malware binaries.

- Another concept that was briefly explored during this thesis was the use of game theory for the modelling of node interactions in networks. Game theory can be used to model the of players within a game based on their prospective gains and losses for making a particular move. Similarly, the cost functions of particular inter-node actions could be based on the nodes' trust probabilities, such that the *optimal actions* for any given node could be calculated. One possibility is to model trust itself as a game in an energy-constrained network, such as a WSN. A malicious node in this environment would aim to gain as much trust in as little time as possible, with its energy being reduced with every interaction. However, this idea was not fully explored because it is impossible to know, or quantify, exactly the gain of a network attack from the attacker's perspective. [52] has begun to work on this, but as also identified in their conclusions, it is currently difficult to know exactly what a node has to gain or lose, making it impossible to derive accurate cost functions. This is a relatively new area of the trust re-

search landscape, so it may be possible in future to more easily quantify the cost functions involved, once real-world data is available about trust-based networks.

Perhaps the foremost challenge during this research has been to design protocols without access to an abundance of real-world trust network training data. Although realistic training data can be generated, it would naturally be beneficial to compare the accuracy of that generated training data with real-world statistics. The availability of this kind of data is likely to improve over the next few years, particularly once the introduction of distributed network infrastructure in real-world applications necessitates the implementation of security protocols like trust inference.

Bibliography

[1] Culture, Media and Sports Committee, "Connected tech smart or sinister? [ONLINE]", in House of Commons Committee Report, available from https://publications.parliament.uk/pa/cm5803/cmselect/cmcumeds /157/report.html, 2023. Accessed June 2023.

[2] A. Sharfman, P. Cobb, "Families and households in the UK: 2022 [ONLINE]", in Analaysis of UK Consensus 2023, available from https://www.ons.gov.uk/peoplepopulationandcommunity/ birthsdeathsandmarriages/families/bulletins/familiesandhouseholds/2022. Accessed December 2022.

[3] D. Kandris, "Applications of Wireless Sensor Networks [ONLINE]" in Encyclopedia, 2022, available from https://encyclopedia.pub/entry/17294. Accessed February 27, 2023.

[4] X. Luo, H. -H. Chen and Q. Guo, "Semantic Communications: Overview, Open Issues, and Future Research Directions," in IEEE Wireless Communications, vol. 29, no. 1, pp. 210-219, 2022.

[5] N. L. Fantana, T. Riedel, J. Schlick, S. Ferber, J. Hupp, S. Miles, F. Michahelles, S. Svensson, "IoT Applications: Value Creation for Industry," In Internet of Things, pp. 153-206, 2022.

[6] S. Sreenivasan, R. Cohen, E. L'opez, Z. Toroczkai, H. E. Stanley, "Structural bottlenecks for communication in networks", in Physical Review E, vol. 75, no. 3, pp. 036105, 2007.

[7] X. Xu, J. Tang, H. Xiang, "Data Transmission Reliability Analysis of Wireless Sensor Networks for Social Network Optimization", in Recent Ad-

vances in Smart Sensor Networks and 5G, 2022.

[8] B. Bodo, J. K. Brekke, J. H. Hoepman, "Decentralisation: A multidisciplinary perspective," in Internet Policy Review, vol. 10, pp.1-21, 2021.

[9] M. Kanellos, "152,000 Smart Devices Every Minute In 2025: IDC Outlines The Future of Smart Things [ONLINE]", 2016, available from https://www.forbes.com/sites/michaelkanellos/2016/03/03/152000-smart-devices-every-minute-in-2025-idc-outlines-the-future-of-smart-things/. Accessed September 2023.

[10] Y. H. Chang, S. Dadhania, "6G Market 2023-2043: Technology, Trends, Forecasts, Players [ONLINE]", in 6G Market Research Report, available from https://www.idtechex.com/en/research-report/6g-market-2023-2043-technology-trends-forecasts-players/911. Accessed January 2024.

[11] A. K. Koshariya, D. Kalaiyarasi, A. A. Jovith, T. Sivakami, D. Hasan, S. Boopathi, "AI-Enabled IoT and WSN-Integrated Smart Agriculture System," In Artificial Intelligence Tools and Technologies for Smart Farming and Agriculture Practices, pp. 200-218, 2023.

[12] S. S. Ravi, M. Aziz, "Utilization of Electric Vehicles for Vehicle-to-Grid Services: Progress and Perspectives," in Energies, vol. 15, no. 2, pp. 589, 2022.

[13] A. Khan, S. Gupta, S. K. Gupta. "Emerging UAV technology for disaster detection, mitigation, response, and preparedness," in Journal of Field Robotics 39, no. 6, pp. 905-955, 2022.

[14] K. A. B. Ahmad, H. Khujamatov, N. Akhmedov, M. Y. Bajuri, M. N. Ahmad, A. Ahmadian, "Emerging trends and evolutions for smart city healthcare systems," in Sustainable Cities and Society, vol. 80, no. 103695, 2022.

[15] K. Kosling, "Global Data Breaches and Cyber Attacks in January 2024 [ONLINE]", in IT Governance, 2024, available from https://www.itgovernance.co.uk/blog/global-data-breaches-and-acyber-attacks-in-january-2024-29530829012-records-breached. Accessed January 2024.

[16] J. von Neumann, "John von Neumann on Natural and Artificial Software, 1948." re-print in Philosophical Mathematics, pp. 33, 2023.

[17] a. Uhde, "A Short History of Computer Viruses [ONLINE]", 2017, available from https://www.sentrian.com.au/blog/a-short-history-of-computer-viruses. Accessed July 2023.

[18] "Fight back against data breaches," whitepaper, IBM, 2023, available from https://www.ibm.com/reports/data-breach. Accessed January 2024.

[19] S. CC, V. Raychoudhury, G. Marfia, A. Singla, "A survey of routing and data dissemination in Delay Tolerant Networks," in Journal of Network and Computer Applications, vol. 67, pp. 128-146, 2016.

[20] H. Kaiyuan and H. Lijuan, "Reliability Study for Distribution Network Considering Adverse Weather," 6th International Conference on Information Science and Control Engineering (ICISCE), Shanghai, China, pp. 449-453, 2019.

[21] Y. Xiao, N. Zhang, W. Lou and Y. T. Hou, "A Survey of Distributed Consensus Protocols for Blockchain Networks," in IEEE Communications Surveys & Tutorials, vol. 22, no. 2, pp. 1432-1465, 2020.

[22] P. R. Nair and D. R. Dorai, "Evaluation of Performance and Security of Proof of Work and Proof of Stake using Blockchain," 2021 Third International

Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV), Tirunelveli, India, pp. 279-283, 2021.

[23] A. Charapko, A. Ailijiang and M. Demirbas, "Bridging Paxos and Blockchain Consensus," 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Halifax, NS, Canada, pp. 1545-1552, 2018.

[24] D. Ongaro, J. Ousterhout, “In Search of an Understandable Consensus Algorithm”, 2014 USENIX Annual Technical Conference, 2014.

[25] S. Poudel, S. Moh, "Task assignment algorithms for unmanned aerial vehicle networks: A comprehensive survey," in Vehicular Communications, vol. 35, no. 100469, 2022.

[26] A. Abdelmaboud, AIA Ahmed, M. Abaker, T. A. E. Eisa, H. Albasheer, S. A. Ghorashi, F. K. Karim, "Blockchain for IoT Applications: Taxonomy, Platforms, Recent Advances, Challenges and Future Research Directions," in Electronics, vol. 11, no. 4, pp. 630, 2022.

[27] D. Wang, Y. Jia, L. Liang, M. Dong and K. Ota, "A Game for Task Offloading in Reputation-Based Consortium Blockchain Networks," in IEEE Wireless Communications Letters, vol. 11, no. 7, pp. 1508-1512, 2022.

[28] R. Asif, S. R. Hassan, "Shaping the future of Ethereum: Exploring energy consumption in Proof-of-Work and Proof-of-Stake consensus," in Frontiers in Blockchain, vol. 6, 2023.

[29] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System [ONLINE]", 2008, available from https://bitcoin.org/en/bitcoin-paper. Accessed June 2022.

[30] I. Butun, P. Ãsterberg and H. Song, "Security of the Internet of Things: Vulnerabilities, Attacks, and Countermeasures," in IEEE Communications Surveys & Tutorials, vol. 22, no. 1, pp. 616-644, 2022.

[31] Y. He, D. Huang, L. Chen, Y. Ni, X. Ma, "A Survey on Zero Trust Architecture: Challenges and Future Trends," in Securing AI-powered Internet of Things (IoT) Ecosystems, Special Issue, 2022.

[32] X. Hui, G. Z. Jin, M. Liu, "Designing quality certificates: Insights from eBay", in National Bureau of Economic Research, No. 29674, 2022.

[33] A. Josang, R. Ismail, "The Beta Reputation System," in BLED, 2002.

[34] R. Boutahala, M. Ayaida and H. Fouchal, "Reducing Security Overhead in the Context of Connected Vehicles," in GLOBECOM IEEE Global Communications Conference, Rio de Janeiro, Brazil, pp. 4304-4309, 2022.

[35] S. Dadkhah, H. Mahdikhani, P. K. Danso, A. Zohourian, K. A. Truong and A. A. Ghorbani, "Towards the Development of a Realistic Multidimensional IoT Profiling Dataset," 2022 19th Annual International Conference on Privacy, Security & Trust (PST), Fredericton, NB, Canada, 2022, pp. 1-11

[36] P. Wang, L. Song, S. Yu and L. Yang, "Analysis and comparison of FEC and FEC-ARQ protection schemes based on RS and Raptor code," 2010 International Conference on Wireless Communications Signal Processing (WCSP), pp. 1-6, 2010.

[37] J. Singh and J. Singh, "A Comparative Study of Error Detection and Correction Coding Techniques," 2012 Second International Conference on Advanced Computing Communication Technologies, pp. 187-189, 2012.

[38] W. Ryan, "An Introduction to LDPC Codes," in Coding and Signal Processing for Magnetic Recording Systems, book, 2003.

[39] T. Brack et al., "A Survey on LDPC Codes and Decoders for OFDM-based UWB Systems," 2007 IEEE 65th Vehicular Technology Conference - VTC2007-Spring, pp. 1549-1553, 2007.

[40] J. H. Cho, K. Chan, S. Adali, "A Survey on Trust Modelling," in ACM Computing Surveys, vol. 48, issue 2, no. 28, pp. 1-40, 2015.

[41] S. Song, K. Hwang, M. Macwan, "Fuzzy Trust Integration for Security Enforcement in Grid Computing," in IFIP International Conference on Network and Parallel Computing, vol. 3222, pp. 9-21, 2014.

[42] S. He, B. Hollenbeck, D. Proserpio, "The Market for Fake Reviews," in Marketing Science, vol. 41, no. 5, 2022.

[43] L. F. Perrone and S. C. Nelson, "A Study of On-Off Attack Models for Wireless Ad Hoc Networks," 2006 1st Workshop on Operator-Assisted (Wireless Mesh) Community Networks, pp. 1-10, 2006.

[44] S. Abbas, M. Merabti and D. Llewellyn-Jones, "Deterring whitewashing attacks in reputation based schemes for mobile ad hoc networks," 2010 IFIP Wireless Days, pp. 1-6, 2010.

[45] G. Ciccarelli, R. L. Cigno, "Collusion in peer-to-peer systems," in Computer Networks, vol. 55, issue 15, pp. 3517-3532, 2011.

[46] I. -C. Lin, T. -C. Liao, "A Survey of Blockchain Security Issues and Challenges," in International Journal of Network Security, vol. 19, no. 5, pp. 653-659, 2017.

[47] Matlab resource, "Linearly Separable Dataset," available from https://uk.mathworks.com/help/stats/support-vector-machines-for-binary-classification.htm. Accessed June 2023.

[48] M. Gustineli, "A survey on recently proposed activation functions for Deep Learning," preprint in arXiv:2204.02921, 2022.

[49] M. Chaudhary, N. Goyal, A. Benslimane, L. K. Awasthi, A. Alwadain and A. Singh, "Underwater Wireless Sensor Networks: Enabling Technologies for Node Deployment and Data Collection Challenges," in IEEE Internet of Things Journal, vol. 10, no. 4, pp. 3500-3524, 2023.

[50] T. Wu, X. Liu, J. Qin and F. Herrera, "Trust-Consensus Multiplex Networks by Combining Trust Social Network Analysis and Consensus Evolution Methods in Group Decision-Making," in IEEE Transactions on Fuzzy Systems, vol. 30, no. 11, pp. 4741-4753, 2022.

[51] J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," 10th International Conference on Malicious and Unwanted Software (MALWARE), Fajardo, PR, USA, 2015, pp. 11-20, 2015.

[52] S. Vijayalakshmi, S. Bose, G. Logeswari, T. Anitha, " Hybrid defense mechanism against malicious packet dropping attack for MANET using game theory," in Cyber Security and Applications, vol. 1, 2023.

[53] Matlab, "https://uk.mathworks.com/", Accessed January 2024.

[54] S. Naveen and M. R. Kounte, "Key Technologies and challenges in IoT Edge Computing," in Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2019.

[55] A. Dolan, I. Ray and S. Majumdar, "Proactively Extracting IoT Device Capabilities: An Application to Smart Homes," 2020.

[56] D. D. S. Braga, M. Niemann, B. Hellingrath and F. B. D. L. Neto, "Survey on Computational Trust and Reputation Models," ACM Computing Surveys,

vol. 51, pp. 1-40, 2018.

[57] S. Song, K. Hwang, R. Zhou and Y. Kwok, "Trusted P2P transactions with fuzzy reputation aggregation," in IEEE Internet Computing, vol. 9, no. 6, pp. 24-34, 2005.

[58] Y. L. Sun, Z. Han, W. Yu and K. J. Ray Liu, "Attacks on Trust Evaluation in Distributed Networks," 2006 40th Annual Conference on Information Sciences and Systems, pp. 1461-1466, 2006.

[59] Ozay, Mete, Inaki Esnaola, Fatos Tunay Yarman Vural, Sanjeev R. Kulkarni, and H. Vincent Poor, "Machine learning methods for attack detection in the smart grid," in IEEE transactions on neural networks and learning systems 27, vol. 8, pp. 1773-1786, 2015.

[60] Junejo, Khurum Nazir, and Jonathan Goh, "Behaviour-based attack detection and classification in cyber physical systems using machine learning," in Proceedings of the 2nd ACM international workshop on cyber-physical system security, pp. 34-43. 2016.

[61] F. Li and J. Wu, "Hit and Run: A Bayesian Game Between Malicious and Regular Nodes in MANETs," 2008 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, pp. 432-440, 2008.

[62] Y. Ruan and A. Durresi, "A survey of trust management systems for online social communities; Trust modeling, trust inference and attacks," in Knowledge-Based Systems, 2016.

[63] J. Sabater and C. Sierra, "Review on Computational Trust and Reputation Models," in Artificial Intelligence Review, 2005.

[64] Gulshan Kumar, "Denial of service attacks - an updated perspective, Systems Science & Control Engineering," 4:1, pp. 285-294, 2016.

[65] G. S. Dhunna and I. Al-Anbagi, "A Low Power Cyber-Attack Detection and Isolation Mechanism for Wireless Sensor Network," in IEEE 86th Vehicular Technology Conference (VTC-Fall), 2017.

[66] M. M. I. M. I. I. Z. M. H. M. Hasan, "Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches," Internet of Things, vol. 7, 2019.

[67] Weizhi Meng, Kim-Kwang Raymond Choo, Steven Furnell, Athanasios V. Vasilakos, and Christian W. Probst. "Towards Bayesian-based trust management for insider attacks in healthcare software-defined networks," in IEEE Transactions on Network and Service Management 15, no. 2, 761-773, 2018.

[68] Ruan, Yefeng, and Arjan Durresi, "A survey of trust management systems for online social communitiesâtrust modeling, trust inference and attacks," in Knowledge-Based Systems, vol. 106, pp. 150-163, 2016.

[69] V. Desnitsky, "Approach to Machine Learning based Attack Detection in Wireless Sensor Networks," 2020 International Russian Automation Conference (RusAutoCon), 2020, pp. 767-771.

[70] K. Ghanem, F. J. Aparicio-Navarro, K. G. Kyriakopoulos, S. Lambotharan and J. A. Chambers, "Support Vector Machine for Network Intrusion and Cyber-Attack Detection," in Sensor Signal Processing for Defence Conference (SSPD), 2017.

[71] J. Xiong et al., "Enhancing Privacy and Availability for Data Clustering in Intelligent Electrical Service of IoT," in IEEE Internet of Things Journal, vol. 6, no. 2, pp. 1530-1540, 2019.

[72] Liang Liu, Zuchao Ma, Weizhi Meng, "Detection of multiple-mix-attack malicious nodes using perceptron-based trust in IoT networks," in Future Generation Computer Systems, vol. 101, pp. 865-879, 2019.

[73] A. Josang and R. Ismail, "The Beta Reputation System," in 5th Bled Electronic Commerce Conference, 2002.

[74] J. Zhang, K. Zheng, D. Zhang and B. Yan, "AATMS: An Anti-Attack Trust Management Scheme in VANET," in IEEE Access, vol. 8, pp. 21077-21090, 2020.

[75] I. T. Javed, K. Toumi, F. Alharbi, T. Margaria, and N. Crespi, "Detecting Nuisance Calls over Internet Telephony Using Caller Reputation," Electronics, vol. 10, no. 3, p. 353, Feb. 2021.

[76] F. Ahmad, F. Kurugollu, A. Adnane, R. Hussain and F. Hussain, "MARINE: Man-in-the-Middle Attack Resistant Trust Model in Connected Vehicles," in IEEE Internet of Things Journal, vol. 7, no. 4, pp. 3310-3322, April 2020.

[77] G. Arulkumaran, R. K. Gnanamurthy, "Fuzzy Trust Approach for Detecting Black Hole Attack in Mobile Adhoc Network," in Mobile Network Applications, vol. 24, pp. 386-393, 2019.

[78] A. Gruner, A. Muhle, T. Gayvoronskaya, C. Meinel, "A Quantifiable Trust Model for Blockchain-based Identity Management," IEEE International Conference on Internet of Things (iThings), 2018.

[79] K. A. Awan, I. Ud Din, A. Almogren, and H. Almajed, "AgriTrust - A Trust Management Approach for Smart Agriculture in Cloud-based Internet of Agriculture Things," Sensors, vol. 20, no. 21, p. 6174, Oct. 2020.

[80] K. Rabadiya, A. Makwana, S. Jardosh, "Revolution in networks of smart objects: Social Internet of Things," in Conference: 2017 International Con-

ference on Soft Computing and its Engineering Applications, Dec. 2017.

[81] O. A. Wahab, R. Cohen, J. Bentahar, H. Otrok, A. Mourad, G. Rjoub, "An endorsement-based trust bootstrapping approach for newcomer cloud services," in Information Sciences, vol. 527, pp. 159-175, 2020.

[82] S. Ji, H. Ma, Y. Liang, H. Leung, Chunjin Zhang, "A whitelist and blacklist-based co-evolutionary strategy for defensing against multifarious trust attacks," in Applied Intelligence, vol. 47, pp. 1115-1131, 2017.

[83] M. Faisal, S. Abbas, H. U. Rahman, "Identity attack detection system for 802.11-based ad hoc networks," in EURASIP Journal on Wireless Communications and Networking, 2018.

[84] X. Meng, "speedTrust: a super peer-guaranteed trust model in hybrid P2P networks," in The Journal of Supercomputing, vol. 74, pp. 2553-2580, June 2018.

[85] M. Wang, C. Qian, X. Li and S. Shi, "Collaborative Validation of Public-Key Certificates for IoT by Distributed Caching," IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, pp. 847-855, 2019.

[86] A. Ugur, "Manipulator: A Novel Collusion Attack on Trust Management Systems in Social IoT," in Proceedings of 10th Computer Science On-line Conference, Vol. 1, pp.578-592, 2021.

[87] W. Fang, W. Zhang, Y. Yang, Y. Liu, and W. Chen, "A resilient trust management scheme for defending against reputation time-varying attacks based on beta distribution," in Science China: Information Sciences, April 2017.

[88] R. Casadeia, A. Aldinib and M. Virolia, "Towards attack-resistant Aggregate Computing using trust mechanisms," in Science of Computer Programming, vol. 167, Aug. 2018.

[89] D. Velusamy, G. K. Pugalendhi, "Fuzzy integrated Bayesian Dempster-Shafer theory to defend cross-layer heterogeneity attacks," in Information Sciences, vol. 479, pp. 542-566, 2019.

[90] A. Amouri, V. T. Alaparthy, S. D. Morgera, "A Machine Learning Based Intrusion Detection System for Mobile Internet of Things," in Sensors, vol. 20, Jan. 2020.

[91] H. Xia, S. Zhang, Y. Li, Z. Pan, X. Peng and X. Cheng, "An Attack-Resistant Trust Inference Model for Securing Routing in Vehicular Ad Hoc Networks," in IEEE Transactions on Vehicular Technology, vol. 68, no. 7, pp. 7108-7120, July 2019.

[92] K. A. Awan, I. U. Din, A. Almogren, H. Almajed, I. Mohiuddin and M. Guizani, "NeuroTrust -Artificial Neural Network-based Intelligent Trust Management Mechanism for Large-Scale Internet of Medical Things," in IEEE Internet of Things Journal, 2020.

[93] D. Mehetre, E. Roslin and S. Wagh, "Detection and prevention of black hole and selective forwarding attack in clustered WSN with Active Trust," in Cluster Computing, vol. 22, Jan. 2019.

[94] V. Busi Reddy, S. Venkataraman and A. Negi, "Communication and Data Trust for Wireless Sensor Networks Using D-S Theory," in IEEE Sensors Journal, vol. 17, no. 12, pp. 3921-3929, June 2017.

[95] W. Yong-hao, "A Trust Management Model for Internet of Vehicles," in Proceedings of the 2020 4th International Conference on Cryptography, Security and Privacy, pp. 136-140, January 2020.

[96] W. Fang, N. Cui, W. Chen, W. Zhang and Y. Chen, "A Trust-Based Security System for Data Collection in Smart City," in IEEE Transactions on Indus-

trial Informatics, vol. 17, no. 6, pp. 4131-4140, June 2021.

[97] J. You, J. Shangguan, L. Zhuang, N. Li, "An Autonomous Dynamic Trust Management System with Uncertainty Analysis," in An Autonomous Dynamic Trust Management System with Uncertainty Analysis: Knowledge-Based Systems, vol. 161, 2018.

[98] S. Nie, "A novel trust model of dynamic optimization based on entropy method in wireless sensor networks," in Cluster Computing, vol. 22, pp. 11153-11162, 2019.

[99] V. Suryani, V. Sulistyo, W. Widyawan, "Two-phase security protection for the Internet of Things object," in Journal of Information Processing Systems Vol. 14, No. 6, pp. 1431-1437, Dec. 2018.

[100] San-shun Zhang, Shi-wen Wang, Hui Xia, Xiang-guo Cheng, "An attack-Resistant Reputation Management System For Mobile Ad Hoc Networks," in Procedia Computer Science, vol. 147, pp. 473-479, 2019.

[101] W. Li, W. Meng, L. Kwok, "SOOA: Exploring Special On-Off Attacks on Challenge-Based Collaborative Intrusion Detection Networks," in the 12th International Conference on Green, Pervasive and Cloud Computing, vol. 10232, pp. 402-415, May 2017.

[102] J. Caminha, A. Perkusich and M. Perkusich, "A smart middleware to detect on-off trust attacks in the Internet of Things," IEEE International Conference on Consumer Electronics (ICCE), pp. 1-2, 2018.

[103] J. Caminha, A. Perkusich, M. Perkusich, "A Smart Trust Management Method to Detect On-Off Attacks in the Internet of Things", in Security and Communication Networks, vol. 2018.

[104] J. Fan, Q. Li and G. Cao, "Privacy Disclosure through Smart Meters: Reactive Power Based Attack and Defense," 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pp. 13-24, 2017.

[105] W. Fang, M. Xu, C. Zhu, W. Han, W. Zhang and J. J. P. C. Rodrigues, "FETMS: Fast and Efficient Trust Management Scheme for Information-Centric Networking in Internet of Things," in IEEE Access, vol. 7, pp. 13476-13485, 2019.

[106] A. Meena Kowshalya, M. L. Valarmathi, "Trust Management in the Social Internet of Things," in Wireless Personal Communications: An International Journal, vol. 96, pp. 2681-2691, Sept. 2017.

[107] F. Khedima, N. Labraouia, A. A. A. Ari, "A cognitive chronometry strategy associated with a revised cloud model to deal with the dishonest recommendations attacks in wireless sensor networks," in Journal of Network and Computer Applications, vol. 123, pp. 42-56, Dec. 2018.

[108] C. V. Mendoza, J. H. Kleinschmidt, "A Distributed Trust Management Mechanism for the Internet of Things Using a Multi-Service Approach," in Wireless Personal Communications: An International Journal, vol. 103, pp. 2501-2513, Dec. 2018.

[109] M. D. Alshehri, F. K. Hussain, O. K. Hussain, "Clustering-Driven Intelligent Trust Management Methodology for the Internet of Things (CITM-IoT)," in Mobile Network Applications, vol. 23, pp. 419-431, June 2018.

[110] X. Liu, Y. Liu, A. Liu and L. T. Yang, "Defending ON-OFF Attacks Using Light Probing Messages in Smart Sensors for Industrial Communication Systems," in IEEE Transactions on Industrial Informatics, vol. 14, no. 9, pp. 3801-3811, Sept. 2018.

[111] E. P. K. Gilbert, B. Kaliaperumal, E. B. Rajsingh, M. Lydia "Trust based data prediction, aggregation and reconstruction using compressed sensing for clustered wireless sensor networks," in Computers & Electrical Engineering, vol. 72, pp. 894-909, Nov. 2018

[112] R. E. Navas, H. L. Boulder, N. Cuppens, F. Cuppens, G. Z. Papadopoulos, "Demo: Do Not Trust Your Neighbors! A Small IoT Platform Illustrating a Man-in-the-Middle Attack," in International Conference on Ad-Hoc Networks and Wireless, vol. 11104, Aug. 2018.

[113] Y. Fu and Z. He, "Bayesian-Inference-Based Sliding Window Trust Model Against Probabilistic SSDF Attack in Cognitive Radio Networks," in IEEE Systems Journal, vol. 14, no. 2, pp. 1764-1775, June 2020.

[114] W. Fang, C. Zhu, W. Chen, W. Zhang and J. J. P. C. Rodrigues, "BDTMS: Binomial Distribution-based Trust Management Scheme for Healthcare-oriented Wireless Sensor Network," 2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC), pp. 382-387, 2018.

[115] F. Zhao, S. Li, J. Feng, "Securing Cooperative Spectrum Sensing against DC-SSDF Attack Using Trust Fluctuation Clustering Analysis in Cognitive Radio Networks," in Wireless Communications and Mobile Computing, 2019.

[116] J. Jia and Y. Li, "A trust attack detection algorithm based on improved K-means clustering," 2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC), pp. 1996-2004, 2020.

[117] S. Ahmed, M. U. Rehman, A. Ishtiaq, S. Khan, A. Ali and Shabana Begum, "VANSec: Attack-Resistant VANET Security Algorithm in Terms of Trust

Computation Error and Normalized Routing Overhead," in Cost-Effective Techniques for Sensors Technology, 2018.

[118] S. G. Fatima, S. K. Fatima, S. I. A. Sattar, "A Security Scheme based on Trust Attack in MANET", in International Journal of Advanced Research in Engineering and Technology, vol. 10, April 2019.

[119] S. Xie, Z. Zheng, W. Chen, J. Wu, H. Dai, M. Imran, "Blockchain for cloud exchange: A survey," in Computers and Electrical Engineering, vol. 81, 2020.

[120] H. Alhumud, M. Zohdy, D. Debnath, and R. Olawoyin, "Cooperative Spectrum Sensing for Cognitive Radio-Wireless Sensors Network Based on OR Rule Decision to Enhance Energy Consumption in Greenhouses", Wireless Sensor Network, 2019.

[121] Weidong Fang, Wuxiong Zhang, Wei Chen, Tao Pan, Yepeng Ni and Yinxuan Yang, "Trust-Based Attack and Defense in Wireless Sensor Networks: A Survey," in Wireless Communications and Mobile Computing, 2020.

[122] M. H. Junejo, A. Rahman, R. Shaikh, K. M. Yusof, I. Memon, H. Fazal, D. Kumar, "A Privacy-Preserving Attack-Resistant Trust Model for Internet of Vehicles Ad Hoc Networks," in Sci. Program, 2020.

[123] Long, H. J., L. Y. Wang, J. Shen, M. X. Wu, and Z. B. Jiang, "Product service system configuration based on support vector machine considering customer perception," International journal of production research 51, no. 18, 2013.

[124] M. Chafii, L. Bariah, S. Muhaidat and M. Debbah, "Twelve Scientific Challenges for 6G: Rethinking the Foundations of Communications Theory," in

IEEE Communications Surveys & Tutorials, vol. 25, no. 2, pp. 868-904, Secondquarter 2023.

[125] A. K. M. B. Haque, B. Bhushan, G. Dhiman, "Conceptualizing smart city applications: Requirements, architecture, security issues, and emerging trends.", in Expert Systems, vol. 39, no. 5, 2022.

[126] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” whitepaper, 2009.

[127] R. M. R. Yasaweerasinghelage, M. Staples, I. Weber, "Predicting Latency of Blockchain-Based Systems Using Architectural Modelling and Simulation", in IEEE International Conference on Software Architecture, 2017.

[128] “Running an Ethereum Node - EthHub.” [online] Available at: https://docs.ethhub.io/using-ethereum/running-an-ethereum-node/ [Accessed 21 December 2020]

[129] D. Gruber, W. Li, G. Karame, “Unifying Lightweight Blockchain Client Implementations,” 2018.

[130] M. Kumar, K. Dutta, “LDAT: LFTM based data aggregation and transmission protocol for wireless sensor networks,” in Journal of Trust Management, vol. 3, 2016.

[131] B. Chandrasekaran, S. Gangadhar and J. M. Conrad, "A survey of multisensor fusion techniques, architectures and methodologies," in SoutheastCon, Concord, NC, USA, 2017, pp. 1-8, 2017.

[132] S. Popov, “The Tangle,” whitepaper, version 1.4, 2018.

[133] K. Kanemura, K. Toyoda, T. Ohtsuki, “Design of privacy-preserving mobile Bitcoin client based on γ-deniability enabled bloom filter,” 2017 IEEE

28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), pp. 1-6, 2017.

[134] Z. Tu and S. Zhang, "Overview of LDPC Codes," 7th IEEE International Conference on Computer and Information Technology (CIT 2007), Aizu-Wakamatsu, Japan, pp. 469-474, 2007.

[135] M. A. Al-Shareeda, S. Manickam, "Man-in-the-Middle Attacks in Mobile Ad Hoc Networks (MANETs): Analysis and Evaluation" Symmetry 14, no. 8, 2022.

[136] J. Rinne, “Majority selection and block-based selection diversity reception methods for 8k DVB-T in a mobile environment,” in 10th European Signal Processing Conference, pp. 1-4, 2000.

[137] Z. Jiang, Z. Zhang, X. Fan, and R. Liu, "Towards All Weather and Unobstructed Multi-Spectral Image Stitching: Algorithm and Benchmark," In Proceedings of the 30th ACM International Conference on Multimedia (MM ’22), Association for Computing Machinery, pp. 3783-3791, 2022.

[138] "eBay", ebay.co.uk. Accessed March 2024.

[139] X. Hui, G. Z. Jin and M. Liu, "Designing Quality Certificates: Insights from eBay," working paper, 2022.

[140] A. Singh et al, “Public Blockchains Scalability: An Examination of Sharding and Segregated Witness,” in Advances in Information Security, vol. 79, 2020.

[141] D. J. Hand, "Trustworthiness of Statistical Inference", in Journal of the Royal Statistical Society Series A: Statistics in Society, vol. 185, no. 1, pp. 329-347, 2022.

[142] J. Wang, Z. Yan, H. Wang, T. Li and W. Pedrycz, "A Survey on Trust Models in Heterogeneous Networks," in IEEE Communications Surveys & Tutorials, vol. 24, no. 4, pp. 2127-2162, 2022.

[143] L. Bilge, T. DumitraÅ, "Before we knew it: an empirical study of zero-day attacks in the real world.", in Proceedings of the 2012 ACM conference on Computer and communications security, pp 833-844, 2012.

[144] W. Duo, M. Zhou and A. Abusorrah, "A Survey of Cyber Attacks on Cyber Physical Systems: Recent Advances and Challenges," in IEEE/CAA Journal of Automatica Sinica, vol. 9, no. 5, pp. 784-800, 2022.

[145] Y. Chae, L. C. DiPippo and Y. L. Sun, "Trust Management for Defending On-Off Attacks," in IEEE Transactions on Parallel and Distributed Systems, vol. 26, no. 4, pp. 1178-1191, 2015.

[146] Y. Cho, "Intelligent On-Off Web Defacement Attacks and Random Monitoring-Based Detection Algorithms," Electronics, vol. 8, no. 11, p. 1338, 2019.

[147] H. Moudoud, Z. Mlika, L. Khoukhi and S. Cherkaoui, "Detection and Prediction of FDI Attacks in IoT Systems via Hidden Markov Model," in IEEE Transactions on Network Science and Engineering, vol. 9, no. 5, pp. 2978-2990, 2022.

[148] R. M. Carnier, Y. Li, Y. Fujimoto and J. Shikata, "Exact Markov Chain of Random Propagation of Malware With Network-Level Mitigation," in IEEE Internet of Things Journal, vol. 10, no. 12, pp. 10933-10947, 2023.

[149] D. Kobiela, D. Krefta, W. Krol, P. Weichbroth, "ARIMA vs LSTM on NASDAQ stock exchange data", in Procedia Computer Science, vol. 207, no. 1877-0509, pp. 3836-3845, 2022.

[150] S. Coleri, M. Ergen, A. Puri and A. Bahai, "Channel estimation techniques based on pilot arrangement in OFDM systems," in IEEE Transactions on Broadcasting, vol. 48, no. 3, pp. 223-229, 2002.

[151] D. Stewart, "Advogato Online Open Source Community 2000-2001", in Mich: Inter-university Consortium for Political and Social Research [distributor], 2011.

[152] D. Li, L. Li, X. Li, Z. Ke, Q. Hu, "Smoothed LSTM-AE: A spatio-temporal deep model for multiple time-series missing imputation", in Neurocomputing, vol. 411, pp. 351-363, 2020.

[153] Zheng Zhang and C. N. Manikopoulos, "Detecting denial-of-service attacks through feature cross-correlation," 2004

[154] A. B. de Neira, B. Kantarci, M. Nogueira, "Distributed denial of service attack prediction: Challenges, open issues and opportunities", in Computer Networks, vol. 222, no. 109553, 2023.

Appendix: A. System Specification and Simulation Software

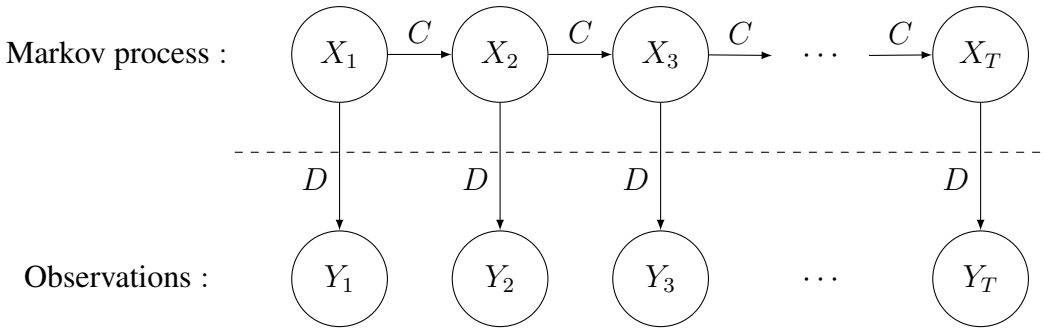
The results produced in this thesis have been generated using Matlab [53], versions 2019a-2023b. The system specifications are as follows:

Table A.1: System specifications

Device	Dell XPS 15 9570
Year	2018
Processor	Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz 2.21 GHz
GPU	NVIDIA GeForce GTX 1050 Ti

Appendix: B. Full Description of Baum Welch Algorithm

Let the HMM be defined as $\theta = (C, D, \pi)$, where C is the transition probability matrix $\mathbb{P}(X_t = j|X_{t-1} = i)$, D is the emission probability matrix $D = d_j(y_i) = \mathbb{P}(Y_t = y_i|X_t = j)$, and π represents the prior state probabilities $\pi_i = \mathbb{P}(X_1 = i)$. Note that i and j represent the possible values of the state at time t X_t and the emission at time t Y_t respectively.



The forward algorithm calculates the probability of seeing the observation set $(y_1, ..., y_t)$ whilst in state i at time t. Let $\epsilon_i(t) = \mathbb{P}(Y_1 = y_1, ..., Y_t = y_t, X_t = i|\theta)$, the forward recursion becomes:

$$\epsilon_i(1) = \pi_i d_i(y_1), \tag{B.1}$$

$$\epsilon_i(t+1) = b_i(y_{t+1}) \sum_{j=1}^{N} \epsilon_j c_{ji}. \tag{B.2}$$

The backward algorithm recursively finds the probability of seeing the observation set $(y_{t+1}, ..., y_T)$, given that the initial state was i at time t. Letting $\phi_i(t) =$

$\mathbb{P}(Y_{t+1} = y_{t+1}, ..., Y_T = y_T | X_t = i, \theta)$, then the backward recursion is:

$$\phi_i(T) = 1, \tag{B.3}$$

$$\phi_i(t) = \sum_{j=1}^{N} \phi_j(t+1) c_{ij} d_j(y_{t+1}). \tag{B.4}$$

This can be rearranged using Bayes' theorem to give the probability of, at time t and given the observation sequence Y, being in the state i:

$$\begin{aligned} \gamma_i(t) &= \mathbb{P}(X_t = i | Y, \theta) \\ &= \frac{\mathbb{P}(X_t = i, Y|\theta)}{\mathbb{P}(Y|\theta)} \\ &= \frac{\epsilon_i(t)\phi_i(t)}{\sum_{j=1}^{N} \epsilon_j(t)\phi_j(t)}. \end{aligned} \tag{B.5}$$

Secondly. we can also deduce the probability of transitioning from state i to j, from time t to time $t+1$ respectively, given Y and θ.

$$\begin{aligned} \xi_{ij}(t) &= \mathbb{P}(X_t = i, X_{t+1} = j | Y, \theta) \\ &= \frac{\mathbb{P}(X_t = i, X_{t+1} = j, Y|\theta)}{\mathbb{P}(Y|\theta)} \\ &= \frac{\epsilon_i(t) c_{ij} \phi_j(t+1) d_j(y_{t+1})}{\sum_{k=1}^{N} \sum_{w=1}^{N} \epsilon_k(t) c_{kw} \phi_w(t+1) d_w(y_{t+1})}. \end{aligned} \tag{B.6}$$

Iteratively, $\gamma_i(t)$ and $\xi_{ij}(t)$ can be used to update the Markov model parameters $\theta* = (C*, D*, \pi*)$ as follows.

The updated probability of being in state i at time 1 is given by:

$$\pi_i* = \gamma_i(1). \tag{B.7}$$

The updated probability of transitioning away from state i becomes:

$$\overset{*}{,}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)}. \tag{B.8}$$

Finally, the probability of output observations being equal to a particular observation v_k is given by

$$d *_i (v_k) = \frac{\sum_{t=1}^{T} 1_{y_t=v_k} \gamma_i(t)}{\sum_{t=1}^{T} \gamma_i(t)}, \tag{B.9}$$

where $1_{y_t=v_k}$ represents the indicator function

$$1_{y_t=v_k} = \begin{cases} 1, & \text{if } y_t = v_k \\ 0, & \text{otherwise.} \end{cases} \tag{B.10}$$

These updates are performed until a satisfactory level of convergence has been achieved.

Appendix: C. Long Short Term Memory

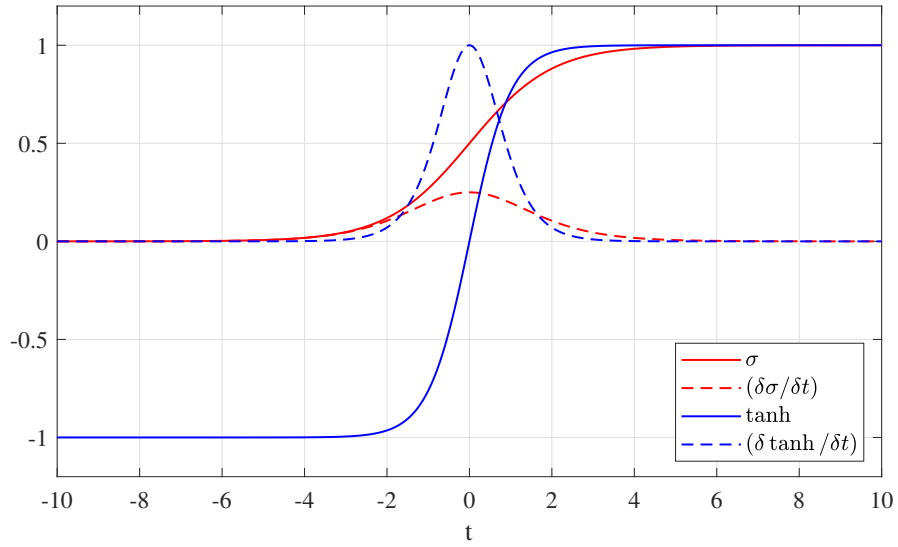


Figure C.1: Comparison of activation functions and their derivative forms.

For example, a common activation function is the sigmoid function, as shown in Fig. C.1, and the derivative of this is always much less than 1. With l layers, the derivative of the activation function is raised to the l^{th} power, resulting in vanishingly small weights being chosen.

In contrast, the LSTM learner is robust to this issue up to a higher number of layers, because it utilises the $\tanh$ activation function to regulate the magnitude of the gradients, such that it can retain a non-zero magnitude for many more multiplications of the activation function derivative than the sigmoid. The $\tanh$ derivative is also plotted against the σ derivative in Fig. C.1.

The LSTM utilises a combination of the sigmoid (σ) and $\tanh$ functions to continually update its *cell state*, i.e., its memory. The procedure involved is visualised as a block diagram in Fig. C.2. The hidden state at time t h_t represents the short-term memory of the RNN, whilst the cell state c_t represents the long-term memory.

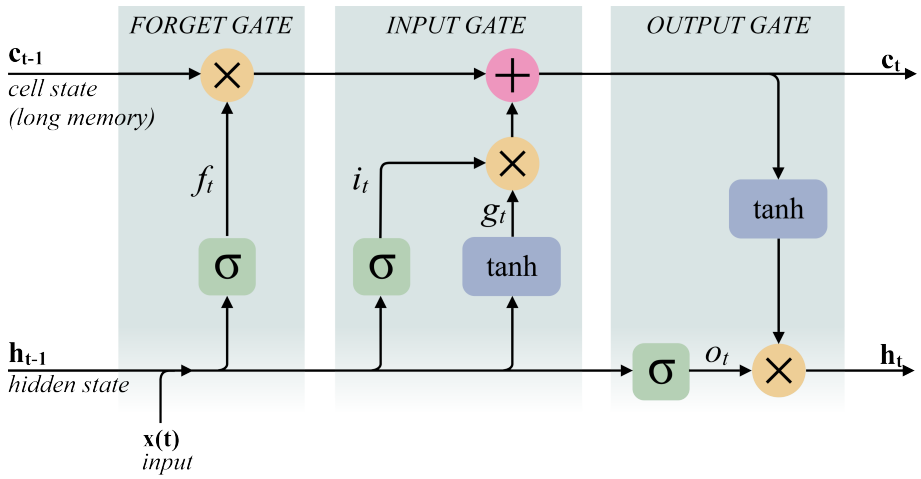


Figure C.2: LSTM block diagram.

The algorithm can be broken down into three gates: the forget gate, the input gate and the output gate. With respect to Fig. C.2, the forget gate f_t decides how much of the previous cell state to *forget*, by multiplying those elements by zero. Its function can be written as

$$f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + b_f\right). \tag{C.1}$$

The input gate i_t determines how much of the new input x_t to commit to the current cell state,

$$i_t = \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + b_i\right), \tag{C.2}$$

The intermediate function g_t scales the input data x_t to lie within $[-1, 1]$, in order to prevent the vanishing gradient problem [48], a common issue for RNNs.

$$g_t = \tanh\left(W_{xg}x_t + W_{hg}h_{t-1} + b_g\right). \tag{C.3}$$

Finally, the output gate o(t) decides how much of the updated short-term memory

to commit to the updated cell state,

$$o_t = \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + b_o\right). \tag{C.4}$$

Finally, the new cell and hidden states are updated,

$$c_t = f_t \cdot c_{t-1} + i_t \cdot g_t, \tag{C.5}$$

$$h_t = o_t \cdot \tanh(c_t). \tag{C.6}$$

Appendix: D. Smoothing Functions

D.1 Moving Average Smoothing

Given n data points $\{j_1, j_2, ...j_n\}$, and a window size of k, the moving average is given by

$$\mu_{MA} = \frac{j_{n-k+1} + j_{n-k+2} + ... + j_n}{k} = \frac{1}{k} \sum_{i=n-k+1}^{n} j_i, \tag{D.1}$$

and, in the next average, the incremented range $\{n - k + 2, n + 1\}$ would be considered, and so on.

D.2 Exponential Smoothing

The smoothed output s_t of time series data j_t at time t can be given by

$$\begin{aligned} s_0 &= j_0 \\ s_1 &= \epsilon j_t + (1 - \epsilon) j_{t-1}, \end{aligned} \tag{D.2}$$

where ϵ is the smoothing factor. Letting represent the data sampling time interval and Θ represent the time constant,

$$\epsilon = 1 - e^{\frac{-\Delta T}{\Theta}}. \tag{D.3}$$

Note that the time constant, Θ, is the time it would take for the smoothing function to reach 68% of its final value when applied to a step function.

Where $\Delta T \ll \Theta$, the smoothing factor $\epsilon \approx \frac{\Delta T}{\Theta}$.

Appendix: E. LDPC Simulation Configuration and Pseudocode

Algorithm 3 Pseudocode for Multi-bit Data Model Simulation

Choose $SNR, t\bar{a}u$, LDPC parameters $(prototype, block\_size)$
for `each packet j` **do**
  Data $X\_j = rand(packet\_length)s$
  for `each node i` **do**
    Encoded data $\leftarrow LDPC\_encode(data, prototype, block\_size)$
    Transmission data $\leftarrow QPSK\_modulate(encoded\_data)$
    Error scope $\in \{bitwise, wholepacket\}$
    Received data $\leftarrow AWGN\_channel(transmission, tau(i), error\_scope, SNR)$
    Demodulated data $\leftarrow demodulate(received\_data)$
    $Y_i \leftarrow decode(demodulated\_data)$
  end for
  Compile received reports $\bar{Y} = [Y_1, Y_2, ...Y_n]$
  $\hat{X} \leftarrow trust\_based\_aggregator(\bar{Y})$, multi-bit or bit-by-bit
  $BER\_j \leftarrow sum(count(\hat{X} - X))/total\_bits$
end for
$BER\_mean = sum(BER\_j)/packet\_count$ =0

Listing E.1: LDPC encoder configuration as used in Chapter 4

```
p = [16,17,22,24,9,3,14,-1,4,2,7,-1,26,-1,2,-1,21,
-1,1,0,-1,-1,-1,-1;
25,12,12,3,3,26,6,21,-1,15,22,-1,15,-1,4,-1,
-1,16,-1,0,0,-1,-1,-1;
25,18,26,16,22,23,9,-1,0,-1,4,-1,4,-1,8,23,11,
-1,-1,-1,0,0,-1,-1;
9,7,0,1,17,-1,-1,7,3,-1,3,23,-1,16,-1,-1,21,-1,
0,-1,-1,0,0,-1;
24,5,26,7,1,-1,-1,15,24,15,-1,8,-1,13,-1,13,-1,
11,-1,-1,-1,-1,0,0;
2,2,19,14,24,1,15,19,-1,21,-1,2,-1,24,-1,3,-1,
2,1,-1,-1,-1,-1,0];

block_size = 27;

pc_matrix = ldpcQuasiCyclicMatrix(blockSize, P);
```