A BLOCKCHAIN-BASED IOT TRUST MODEL

by

Sarah Asiri

BSc Information Systems, Al-Imam Muhammad Ibn Saud Islamic University, Saudi Arabia, 2013

A thesis
presented to Ryerson University
in partial fulfillment of the
requirements for the degree of
Master of Science
in the program of
Computer Science

Toronto, Ontario, Canada, 2018

AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A THESIS

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my thesis may be made electronically available to the public.

A Blockchain-Based IoT Trust Model

Master of Science 2018

Sarah Asiri

Computer Science

Ryerson University

Abstract

The Internet of Things (IoT) is a heterogeneous network of interconnected objects or 'things' that are typically connected via the Internet. Trust in most IoT networks is presumed implicitly. This implicit trust assumption can be abused by adversaries to disrupt the network and manipulate reputations of trusted devices. To tackle IoT trust issues, we use permissioned blockchains that utilize Smart Contracts (executable policies) to evaluate and refine IoT devices' trust. Blockchains replicate a permanent append-only record of all transactions occurring on a network on multiple devices. This prevents adversaries from modifying previous transactions to influence trust evaluations. In this thesis, we propose an IoT trust model that uses Blockchains to record and validate IoT devices' identities and dynamically evaluates the trustworthiness of devices in the IoT network. Moreover, our model allows for different levels of security based on the sensitivity of data being transmitted across the IoT network.

Acknowledgements

To start, I would like to thank the Ministry of Higher Education in Saudi Arabia for sponsoring my journey to complete my Master's degree.

My deepest gratitude and thanks goes to my thesis advisor Professor Dr. Ali Miri. For his constant support, advice, guidance and feedback. The door to Dr. Miri's office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this thesis to be my own work, but steered me in the right the direction whenever he thought I needed it.

I would also like to extend my profound gratitude to my parents and to my friends for providing me with unfailing support and continuous encouragement throughout my years of study in Canada and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Dedication

Throughout my life, two special individuals have always been there to support my dreams.

To

Mom & Dad

Gentle and strong souls who taught me to trust in Allah, believe in hard work and encouraged me to believe in myself. I would not be who I am, here today, without their love and support.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Today, the use of the IoT has increased exponentially from industrial applications to smart homes and wearable devices. This increase has also resulted in major security concerns and reported vulnerabilities. One recent attack earlier this year was a large-scale Distributed Denial of Service (DDoS) attack on Dyn, an Internet service provider[1]. This DDoS attack utilized a botnet [2] known as *Mirai*, made of a large number of vulnerable IoT devices, such as IP surveillance cameras, and managed to take down major Internet services such as Twitter, Netflix, PayPal and Amazon. Sensitive and confidential information exchanges over IoT networks have also become targets for attackers. These attacks focus not only on eavesdropping on communication, but also on spreading spam and malware, and launching various malicious actions. Common security defences in use so far have proven ineffective or insufficient against even some basic and unsophisticated attacks on IoT networks and devices. One of the common reasons for this weakness is the implicit presumption of trust between devices that make up these types of networks. Therefore, IoT networks are in need of a mechanism that facilitates operations and secure communication among their devices by constantly evaluating trust and identities of devices on the network.

---

[1] Dyn is an Internet company that is known for its domain registration services: https://dyn.com

[2] Botnets work by infecting large numbers of devices with malware without the knowledge of their owners. Once malware is installed, infected devices are used for malicious tasks, such as launching DDoS attacks.

The rest of this chapter provides a brief introduction to related concepts in IoT. This is followed by our problem statement, and an overview of our contributions made in Section 2. An outline of the thesis can be found in Section 3.

1.1 Trust and The Internet of Things

An IoT network is made up of a combination of interconnected objects or (intelligent) *'things'* that are typically connected via the Internet. IoT devices are often assigned unique IP addresses to help identify them on the network. These devices can often be managed remotely and have the ability to interact with one another. They can also collect or sense data from their surroundings, and process them into useful information used by the system or its users [3]. The integration of the IoT into day-to-day activities has been growing fast, and is becoming more popular in various fields and sectors including healthcare, industrial automation, and manufacturing. Smart IoT devices can integrate smoothly with their surroundings to provide access to various types of information and services in a timely manner. For example, a *smart thermostat* can allow its users to control room temperatures across rooms remotely and a *smart light fixture* can be used to turn on lights before walking into a room. In addition to smart thermostats and smart lights, IoT devices in smart homes can include door locks, surveillance cameras, coffee makers, and refrigerators, many of which can be remotely operated by mobile applications, or bring the latest weather forecast and news to users. Devices in the IoT network can initiate and participate in *transactions*[3] that allow information sharing as part of operational flow in smart environments, such as smart homes. For example, a refrigerator can inquire about the current room temperature from a neighbouring thermostat connected to the same network, and use it to adjust the operation of its cooling unit. The smart refrigerator can access a user's calendar and display it along

---

[3] Any information exchange that occurs on a network is commonly referred to as a *transaction*

with time and weather information on its display. Compromising a smart refrigerator or its communication with other devices on the same network might appear insignificant due to the nature of information it transmits. However, controlling a compromised smart home or eavesdropping on its communication can lead to the capture of users' passwords [4], or all the above can be leveraged to gain physical access to the compromised smart home. The impact of these types of compromises can be more severe. In a recently reported incident [5], hackers were able to gain access to an industrial heating system and manipulate building temperatures. If such attacks are mounted against critical infrastructures such as nuclear plants or data centres, the consequences can be catastrophic.

Many existing IoT security solutions have limited success in minimizing the impact of compromised IoT devices, and their communications. For example, these schemes cannot counter insider Sybil [4] attacks that occur after an authenticated device turns rogue. It is possible to spoof devices' identities or forge new identities to enable adversaries from masquerading and acting as legitimate devices. Devices with forged identities are known as Sybils. Trust models can offer a strong counter-measure against these and other types of attacks, by providing a measurement of trustworthiness to network devices. These models can use techniques similar to those employed in anomaly detection solutions to label a node as *trustworthy*, or *untrustworthy*. This classification of devices is based on comparison of current observed behaviour against past behaviour. Recommendations and feedback from other IoT devices can also provide a peer-to-peer feedback system in the proposed trust model. Important security services required in IoT networks include *authentication* and *authorization*. Authentication is the establishment of identity between communicating parties [6]. IoT devices should be able to verify each others' identities to guarantee that a device is in fact who it is claiming to be. Authorization on the other hand, is ensuring that a certain authenticated device has permission to access a resource or perform a certain task. There is

---

[4] Sybil attacks occur when an adversary replicates or impersonates identities of existing nodes on a network.

currently little work in the literature on utilizing trust models in addressing the IoT security challenges listed above. We will discuss these models in detail in Chapter 2. These solutions separate device authentication from trust evaluation. This separation can potentially offer a vulnerability that an attacker may exploit.

1.2 Problem Statement

Rising adoption of IoT devices in our daily lives makes these devices a desirable target for adversaries. IoT networks are of a dynamic and heterogeneous nature, allowing different (untrusted) devices to join the network for an unspecified amount of time. It is possible for devices' intentions to change during their connection time in the case of compromise, or for malicious devices masquerading as benign [7]. An adversary can also control multiple identities (Sybils) and force negative feedback in order to make a trusted device appear untrustworthy to its peers or an untrustworthy device appear trustworthy. Once an IoT device becomes malicious, it is important to detect it and evaluate its trustworthiness in order to prevent it from affecting the remaining devices in the network. Employing a decentralized setup that is suitable for the heterogeneous nature of the IoT and can promote authentication of devices and verification of their transactions to protect against various types of attacks is of utmost importance. Such a setup also needs to be supported by transparent, tamperproof mechanisms to record and update trust indicators based on (historical) transactions and peer recommendation, while enabling verification of transactions' sources, the integrity of messages, and the reputations of devices.

To tackle the discussed trust abuse problems in the IoT, we propose a dynamic decentralized IoT trust model based on blockchains. This model integrates authentication and authorization together with a dynamic machine learning-based trust evaluation approach. The model limits transactions to authorized IoT devices, and protects against internal attacks as it detects devices that do not meet trust requirements. In addition, our model provides different levels of security depending on the sensitivity of the data being transmitted. This can significantly minimize calculation overhead for transmitting publicly available information. Moreover, the model uses collected device information to predict trust of new devices that have just joined the network. The model can be tuned based on the context of services and information exchanged, using different pre-defined threshold parameters. In a given transaction, if a device fails to meet the related threshold parameter, it is completely ignored by the remaining devices in the IoT network. Hence, compromising a single device will not affect the whole network. To provide a bootstrapped security for our proposed model, we use a customized *permissioned* blockchain to facilitate identity management and record keeping of all transactions on the IoT network. Permissioned blockchains only allow authorized devices to become part of the IoT network. The idea behind blockchains is to store a permanent record of all transactions occurring on a network on all devices connected to that network. Devices on the network share a replica of the same copy, which lists all transactions tracing back to the network's creation. This copy is referred to as a *public ledger*. Any transaction between two devices is appended to the ledger and cannot be modified or forged. This produces secure storage and an immune history of all transactions, preventing adversaries from modifying previous transactions to influence trust evaluations of IoT devices. Information about transactions is available for authorized devices to access at anytime. Maintaining the ledger and evaluation of devices' trust is designated to capable and trusted entities on the network. We have built a proof-of-concept prototype to evaluate the effectiveness of some of the key components in our model.

Part of the work in this thesis has appeared in 'An IoT Trust and Reputation Model Based on Recommender Systems' (in Proceedings of the 14th Annual Conference on Privacy, Security and Trust (PST 2016))[8]. In this paper, we proposed our IoT trust and reputation model that employs distributed Probabilistic Neural Networks (PNNs) to distinguish trustworthy nodes from malicious ones. Our model tackled the cold start problem in IoT environments by predicting ratings for newly joined devices based on their characteristics and learns trust over time. Another manuscript entitled ‘Sybil Resistant IoT Trust Model Using Blockchains’ is also in preparation for submission.

1.3 Thesis Outline

The remaining chapters of this thesis are organized as follows:

Chapter 2 focuses on background information and a literature review. A quick overview of IoT network architectures as it pertains to our models in presented in this chapter. We discuss various security requirements in IoT networks, and conditions that make securing these types of networks more challenging than traditional networks. We list different types of attacks, including Sybil attacks. The IoT Trust model concept is also introduced, as well as a comprehensive analysis of related work in the literature. We also given an introduction to blockchains, and their key components and functionalities, with a particular focus on the permissioned blockchain architecture which we employ as part of our solution.

Chapter 3 is the main chapter of our thesis. It provides a detailed overview of our proposed trust model. Different components of this model are presented and discussed. Our proof-of-concept is based on the open-source tool *Hyperledger Fabric*[5], which can support our peer authentication and feedback. We provide a detailed state and transaction flow in this blockchain implementation, and show how it can be utilized for node enrolment, trust score initialization, and dynamic trust evaluation. We will show how the model achieves its listed security goals.

Chapter 4 provides thesis conclusions and discussions of possible future work.

---

[5] Hyperledger is an open source collaborative effort hosted by The Linux Foundation and built for blockchain solutions’ development in various fields: https://www.hyperledger.org/

Chapter 2

Background and Literature Review

One of IoT's primary purposes is delivering services to users by connecting *'things'*, regardless of their nature, size, operating system or location. Communication and data transmission between connected *things* is favored to be timely and reliable. Until recently, security of many IoT devices has been disregarded by many vendors and manufacturers. The reason behind this is that vulnerabilities are catching up with the minimum security that was implemented in most of IoT devices. An emphasis on securing information transmission is signified in sectors where protecting data is vital, such as military confidential data or patients' private readings from medical devices in healthcare. As mentioned before, IoT devices can be connected to the Internet, this connection can expose devices to external adversaries. For example, security researchers proved that smart cars can be controlled and stopped remotely [9]. Compromising IoT devices in such manner can jeopardize individuals' safety. Relying on authentication schemes for security is insufficient for IoT networks; as such schemes do not protect against attacks launched internally by masqueraded devices. Therefore deploying a mechanism to validate identity and trustworthiness of connected devices is imperative. Evaluating trust can allow for transactions to occur only between trustworthy devices, which can greatly minimize chances of confidential data and network compromise.

This chapter details the challenges of securing IoT environments and trust models by introducing relevant background information for the concepts involved in our proposed model. We also discuss related work that has been introduced in IoT trust models and the machine learning component that enables our model to learn nodes' behaviour. Section 1 of this chapter discusses IoT, its architecture and security requirements and challenges. In Section 2, we examine trust models in IoT context and discuss the role of Probabilist Neural Networks (PNN) in predicting and calculating trust. Analysis of attacks on IoT trust models is discussed in Section 3. Section 4 explores blockchains and their methodology. Blockchains in IoT are discussed in Section 5. Section 6 discusses security of blockchains. Section 7 is a conclusion to this chapter.

2.1 The Internet of Things

IoT devices, also known as *smart devices*, are devices with a capability to connect to the Internet. One of the essential characteristics of IoT is devices' ability to send and receive data that is gathered from surroundings. Collected data is processed for analysis to produce meaningful information and reports. These devices can range from smart home appliances to small wireless sensors. IoT devices and sensors can cover geographically large areas to detect motion, sound or temperature. One common application of IoT is Smart Cities. A smart city is equipped with different types of IoT devices, such as lights, cameras, utility meters, motion and temperature sensors. For example, sensors on roads can help notify drivers of traffic jams during rush hours and suggest alternative travel times. Additionally, deploying sensors in parking facilities can inform drivers of vacancies in parking lots before arriving to the facility. This minimizes search times for available spots in busy parking areas, allowing drivers to save on energy and gas emissions.

The number of attacks that exploit emerging IoT networks is increasing. DDoS attacks that utilize IoT devices are also on the rise. The most recent DDoS attack was launched using the Mirai botnet discussed in Chapter One. Although cryptographic schemes can help minimize the resulting damage of compromising these exposed devices, such schemes fail to prevent against insider attacks that abuse trust. Furthermore, cryptographic schemes can be computationally expensive for IoT devices that can be constrained in terms of memory, bandwidth and processing. Performing heavy computations on these devices results in higher maintenance costs and shorter devices' life spans. Therefore, solutions to secure IoT networks, while taking into consideration devices' limited resources are important. IoT trust models can help secure IoT devices and networks by detecting malicious devices on the network. Devices that are suspected to be malicious are denied communication and ignored by neighbouring devices on the network.

2.1.1 Architecture of IoT

Different implementations and designs of IoT vary based on the application and architecture. In order to obtain a better understanding of the nature of IoT, we discuss its common layered architecture. One of the common IoT architectures is introduced by Jong-Moon Chung [1]. The architecture is composed of four layers. Layers start with IoT devices and sensors that collect data and end with an application that delivers information to its users.

1. Layer 1: Sensor and Edge layer:

   An edge layer in an IoT network contains IoT devices and sensors that can perform light-weight computations and processing. Edge processing is known as fog computing. Hence, fog computing differs from cloud computing in data processing location. In cloud computing processing of data is performed on a remote server. IoT is designed to transmit data in a timely fashion across networks. Collected data is processed

---

[1] The architecture is discussed in an online course: 'Internet of Things & Augmented Reality Emerging Technologies' (https://www.coursera.org/learn/iot-augmented-reality-technologies)

into meaningful information for reporting, analytics or decision making. Limited by available resources, performing all processing on the edge level is infeasible for IoT devices. Therefore, remaining processing is usually pushed to the cloud.

2. Layer 2: Gateway and Network layer:

   This layer is mainly concerned with message routing [10]. IoT gateways can be constructed from several types of networks, including Wide Area Networks (WANs), Mobile Communication Networks, or Local Area Networks (LANs) with Wi-Fi or Ethernet connectivity. In addition to networking capabilities, micro-controllers, signal processing, access points, encryption capabilities and embedded operating systems exist to facilitate external connections to IoT devices and sensors.

   The integration of various network types into the IoT platforms is crucial for maintaining robust performance that can handle huge volumes of data generated by a wide range of heterogeneous sensors and devices across the network with support to different protocols and technologies. Scaling for multiple technologies to support IoT services is key factor for a successful design.

3. Layer 3: Management Service layer:

   Modeling of devices, their configurations and management of data-flows and security controls is completed in the management service layer. Additionally, business rules, analytics and logic can be implemented in this layer. Management of data can be considered of both periodic and aperiodic characteristics. In periodic IoT, data filtering is required, as data will be collected periodically, which generates huge volumes of data that must be filtered for making business decisions. On the other hand, in aperiodic IoT, actions are triggered immediately based on collected readings.

4. Layer 4: Application layer:

In the application layer, processed information is delivered to different actors and used in a variety of applications. Such applications include: industry automation, healthcare, education, transportation, logistics, surveillance and people tracking.

Our proposed trust model is implemented on layers 3 and 4 of the above IoT architecture.

2.1.2 Security in The Internet of Things

As the number of Internet-connected IoT devices increases, they become more vulnerable to external attacks. Compromising a single device on an IoT network can allow adversaries to control the whole network. Therefore, addressing security of the overall network is important.

Security Requirement

IoT networks typically utilize constrained devices that use low-bandwidth standards and must maintain an open secured communication channel with more powerful devices, such as Smartphones or gateways. Guaranteeing this channel's security requires optimal cryptography algorithms and proper key-management systems, as well as security protocols that connect all these devices through the Internet [11] :

- **Authentication:** Authentication can be difficult to achieve in IoT, due to the nature of its' constrained devices and their heterogeneity. Authentication is identity establishment between communicating parties [6]. IoT devices should be able to verify each others' identities to guarantee that an object is in fact who it is claiming to be. In IoT networks, authentication usually occurs between an IoT device and a central authority.
- **Authorization**: Authorization is ensuring that a certain device has appropriate privileges to access a resource or perform a certain task.

- **Identity Management:** Different devices are connected to IoT and the number of these devices is increasing. IoT devices' can be impersonated maliciously by adversaries to perform malicious tasks. Hence, identities of IoT devices on the network require validation.

- **Privacy:** With the increasing acceptance of IoT around us, the amount of data that is generated by IoT devices is large. IoT devices can transmit information that categorizes their users' behaviours, preferences and patterns. In most cases, collected information can be analyzed and used for profiling or even marketing. Therefore, privacy concerns in IoT have been raised.

- **Confidentiality:** All kinds of data travel through the IoT network. Some of that data is confidential. In the context of health-care, for example, almost every packet traveling through an IoT network contains patients' confidential information that allows physicians to check their patients' medical status remotely. Such data should be protected from interception. It is of equal importance as well to protect the data stored on these devices.

- **Integrity:** Integrity refers to assurance that information has not been modified by unintended parties. Preserving messages integrity' is crucial in many IoT applications, such as patients' medical information. Therefore, confidential data should be immune to change throughout the transmission process.

- **Availability:** Data in IoT should be available for access by authorized users at all times, whenever they require it. Intrusion detection and protection against DoS and DDoS attacks are crucial to guarantee a smooth flow of data.

Security Challenges

Securing devices in IoT is not the same as securing a traditional network. It is important to take into consideration several factors that make securing IoT networks more difficult [8]:

- **Dynamic Topology:**

  IoT mostly consists of mobile devices that facilitate control of other connected devices and allow for information retrieval. Devices in IoT networks can join the network for undetermined amounts of time. For example, a smart car can be part of an IoT network for seconds, as it is only passing an access point. On the other hand, some devices join an IoT network permanently. New devices connect and disconnect constantly and the network needs to scales and adapt to these changes without affecting performance. Since our model operates in a decentralized setup, it has the ability to scale and adjust to joining and leaving devices.

- **Recourse Limitations:**

  Applying symmetric cryptographic algorithms to secure resource constrained devices appears inefficient. The majority of IoT devices are very limited in terms of memory, bandwidth, battery and processing power; possible keys for all network nodes cannot be stored on each single node. For that reason, Symmetric Key Cryptography (SKC) is considered an infeasible solution. Similarly, employing Public Key Cryptography (PKC) can possibly overload the devices' capabilities. To tackle this resource limitation in our model, heavy cryptographic processing, key management and transaction validation are handled by capable nodes on the network.

- **Heterogeneous Nature:**

  Devices that join the IoT network have different processing power, operating system, bandwidth, vendors and functionalities. It is challenging to use traditional security measures to fit all devices. Types of devices that can join IoT networks cannot be predicted. For that reason, IoT needs an innovative solution that can be implemented at the edge layer of the IoT network, regardless of the devices' features, to help determine trustworthy devices from those which can be malicious. Our model facilitates RESTful interactions between the model and IoT devices to minimize network overhead. Representational State Transfer (REST) is an architecture for accessing and modifying a resource on the Internet. It is mainly concerned with identifying resources and modifying them with basic Hyper Text Transfer Protocol (HTTP) methods. The protocol is supported by the majority of IoT devices, which allows our model to tackle the heterogeneity of IoT.

2.1.3 Attacks on IoT

IoT networks are distributed by nature with a variety of connected smart *'things'* and sensors. These *'things'* transmits different types of information. Some of the transmitted traffic is information that can be found publicly on the Internet, such as weather information. Similarly, IoT devices can be used to transmit confidential information. This can range from personal information and location to health and medical information. A network implementing a trust model to safely manage nodes and their interactions is a desirable target for attackers. In this section, we analyze possible attacks that can be launched against IoT networks and similarly, IoT trust models. We discuss Sybil attacks in more detail as our work provides a proof-of-concept implementation to protect against them.

Defining Threats

In order to gain an understanding of our model's resilience, we first discuss the nature of possible attacks on trust model solutions. Yu et al. [12] categorizes common attacks launched on trust mechanisms in network setups similar to IoT. Attacks can be divided to two major categories:

1. **External Attacks:**

   External attacks on IoT networks occur when an adversary have no knowledge of the network's cryptographic keys and the attack is launched from outside the network. In such scenario, the adversary attempts to eavesdrop on the communication channel to salvage any useful information either about the content of transmitted information or about the implemented trust model itself and its operations. Upon successfully capturing information, adversaries can launch more attacks such as DoS, replay, message alteration and Sybil attacks.

2. **Internal Attacks:**

   Unlike external attacks, an adversary is assumed to be controlling a trusted entity on a network. Therefore, the attack is launched from within. This form of attack is more difficult to detect as it can occur when a trustworthy device is turned rogue after gaining trust on the network.

Table 2.1 shows a summary of common attacks on IoT networks and their description.

Sybil Attacks

Typical IoT networks are mostly dependent on assumptions of one identity per device. Sybil attacks take advantage of that trust presumption by hijacking vulnerable devices to claim different identities. Sybil attacks allow adversaries to replicate and control forged device identities to influence the IoT network negatively. This can result in faulty reports generated by the IoT network or Spam dissemination [13]. Adversaries can further use these identities to manipulate trust scores and device reputations in an IoT trust model. In consequence,

Table 2.1: Common Attacks on IoT Networks

Attack	Category	Description
DoS	External/Internal	Floods IoT networks with requests in an attempt to prevent it from functioning properly causing network unavailability to legitimate users.
Sniffing Attack	External	Eavesdrops on information transmitted among nodes on the network.
Reply Attacks	External/Internal	Replays messages that are already transmitted or injects fabricated information into them.
Sybil Attacks	External/Internal	Replicates and impersonates identities of existing nodes on the network.
Wormhole Attacks	Internal	Creates a new channel to another part of the network where messages are replayed.

adversaries trick the network into appearing to possess a relatively large influence on the network. Sybil attacks can be labeled and classified based on different factors [14]:

- Nature of communication (Direct or Indirect): If adversaries use fake identities to communicate with trusted nodes directly, this is known as a direct Sybil attack. On the other hand, an indirect attack occurs when the adversary uses his/her legitimate identity (the malicious node) to communicate with a trusted node. [14].
- Attacker's Resources: The number of identities an attacker can control at the same time increases their chances of a successful attack. When several identities join and leave during different times, it makes detection of Sybils harder.

Examples of Sybil Attacks

Sybil attacks can occur in different setups. To gain a better understanding of their impacts, we glance over some of the popular examples where Sybil attacks can be launched.

1. Vehicular Ad-hoc Networks (VANETS) : VANETS are vulnerable to Sybil attacks. VANETs connect nearby cars and treat each car in the network as a node. The priority in these networks is road safety. Compromising these nodes is possible, due to weak authentication implemented on them. It is possible for an adversary to inject malicious code to create several fake identities on the network to disturb traffic or jam it. [15].
2. Voting System: Distributed voting systems are peer-to-peer (p2p) networks that detect malicious nodes or decide how resources are distributed. Once an attacker creates multiple identities, they can vote multiple times according to their needs [14].
3. Tor[2] Networks: Tor networks anatomize traffic through implementing the Onion protocol to protect traffic. A sybil in the Tor network is a node or rely that has multiple identities, acting as a different node in each communication. This allows an attacker to gain large influence in the network and eventually compromise it [16]. Upon successful launch of the attack, more severe attacks can be carried out including tampering traffic to exit nodes, DoS, website fingerprinting and bridge address harvesting.

2.2 IoT Trust Models

To improve on IoT security, different approaches were explored to accommodate the limited resources and capabilities of IoT smart devices and sensors. A mechanism is required to identify trustworthy devices from malicious ones without affecting service delivery. One way to achieve that is through trust and reputation models.

The concept of trust has been around since the beginning of time. Through past experiences, humans have learned to develop and evaluate trust between one another. When

---

[2] Tor is an open network software that helps defend against traffic analysis: https://www.torproject.org/

in doubt, they require consultation from friends or neighbours to make decisions. The crucial thing about trust is that it is built over time. Trust can be defined as the subjective probability by an individual, expecting that another individual's actions are dependent on the first individual's well being [17]. In other words, trust is to expect no harm from that individual. Reputation, on the other hand, is others' opinions and what is believed in a certain standing [18]. In the context of IoT, trust definitions are very similar. Devices function within a community, as all devices are neighbours, and can share opinions about other devices based on their interactions. These opinions help determine if devices are trusted and consequently allow for detection of untrustworthy devices. Implementation of trust and reputation models is a useful security mechanism in environments that involve interaction of various entities [19]. Trust establishment is very critical in IoT. Various scenarios and circumstances exist that highlight the importance of a trust check that can help determine trustworthy devices from malicious ones. Residential smart homes, Hospitals, manufacturers, and military sectors that apply IoT technologies to facilitate their work, deal with confidential information. Furthermore, such IoT environments employ various devices that are dispersed randomly and pose high chances of physical compromise. Therefore, constantly verifying trustworthiness and legitimacy of connected devices in a fast manner is crucial [8].

The number of trust models that have been introduced and implemented in the context of IoT is very limited. Currently, only one model exists in the literature that employs blockchains for trust in Wireless Sensor Networks (WSN). We review this model in Section 4, after discussing blockchains' architecture and functionality. For the remaining of this section, we review three trust models related to IoT and show how our proposed model can tackle many of their shortcomings.

Trust Management (TRM):

Chen et al. [7] proposed TRM, a model to overcome trust establishment problems in Cyber Physical Systems (CPS) devices and wireless sensors, based on reputations in IoT. The model aims at securing Machine to Machine (M2M) communication between the nodes and also protect against malicious node attacks through the use of fuzzy logic to analyze nodes' behaviour and determine trustworthiness [7]. It utilizes Quality of Service (QoS) metrics that analyze attributes such as: packet forwarding rate and packet delivery ratio, which are monitored by neighbouring nodes. Each node independently overhears its neighbouring nodes' packet forwarding activities. This monitoring is related to the proportion of correctly forwarded packets with respect to the total number of packets to be forwarded during a fixed time window. Each node in the network maintains a data forwarding information table. The table includes only the data forwarding transaction information by overhearing neighbouring nodes [7]. In review of this model, we identified that it is limited to devices in CPS, involving only wireless sensors. A model that is not constrained by a specific context can be more effective. The authors rely on data forwarding information to evaluate trust. Yet, no measure is taken to verify the source of that information. This can allow transaction information about nodes' forwarding behaviour to be fabricated and neighbouring nodes can be manipulated into trusting a malicious node. An adversary can analyze a node's packet forwarding behaviour and imitate it to appear trustworthy. Therefore, validating nodes' identities is imperative. Also, the scenario in which a node turns malicious after a period of normal behaviour is not addressed [8].

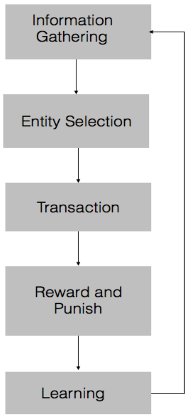


Figure 2.1: Phases of TMS By [1]

Trust Management System (TMS):

Saied et al. proposed TMS for IoT that utilizes nodes' previous experiences, for various services and contexts, to predict trust that can be put into a node for accomplishing a certain task [1]. The model's primary objective is to facilitate cooperation in the heterogeneous IoT architecture for devices with different resource capabilities, to build a community of trusted devices collaborating with one another according to their task [8]. Figure 2.1 shows the phases of their proposed model.

To better understand their setup, functionality of each phase is summarized below [1]:

1. **Information Gathering:**

   At the beginning, all nodes are presumed trustworthy to deal with the cold start problem at the bootstrapping process. Upon completion of a service, the requesting node evaluates the assisting node's behaviour with a positive or negative feedback, where they are stored in the trust management system. Trust ratings are collected for the same node in different contexts. The report sent by the evaluating node includes: service, capability, score and time.

2. **Entity Selection:**

Upon receiving a request, the trust manager 'returns a list of trustworthy assisting nodes to the requester'. Returned candidates that were chosen in prior selection match the requested service. After that, the list is narrowed down for the final selection process. These nodes in the final list pertain context as that of the requesting node. In the selection process, different reports are assigned different weights, as some reports are more important. These weights are used for trust computation. Additionally, a Quality of Recommendation (QR) is calculated for issued reports. Finally, trust is computed using calculated weights and QR. N is the score (positive or negative) in the issued report:

$$T_i = \frac{1}{\sum_{j=1}^{n} w_{R_{ij}}} \times \sum_{j=1}^{n} (w_{R_{ij}} \cdot QR_j \cdot N_j)$$

3. **Transaction:**

   The requesting node receives a list of trustworthy nodes to assist it and chooses one to carry out an interaction.

4. **Reward and Punish:**

   After a transaction is completed, the requesting node either rewards or punishes the selcted node with its report by providing a positive or negative score. This is mostly based on direct observations from the requesting node.

5. **Learning:**

   This phase handles recommendations and reputation updates. Updating of QRs occurs by mapping current evaluations against previous recommendations during selection phase.

The TMS model addresses different attack scenarios, such as bad mouthing attacks and selective attacks. Bad mouthing attacks occur when ratings of nodes are manipulated. An adversary can force bad ratings for certain nodes to deny them service or to ruin their reputation within the community. Yet, we need to be able to verify the source of submitted recommendations. Moreover, protecting integrity of the history of all recommendations is important in order to prevent adversaries from changing previous recommendations, allowing them to manipulate devices' reputations.

Distributed Dynamic Trust Management Protocol (DDTMP):
Bao and Chen [20] proposed a Distributed Dynamic Trust Management Protocol (DDTMP) that takes into consideration the honesty of collected recommendations, cooperativeness of devices and community interests, for a social community of IoT devices that considers social relationships among devices' owners. Each node in the network evaluates trust within a set of nodes that communicate occasionally [20]. The authors consider the case of IoT networks that consist of human-operated/carried smart objects [20]. Hence, 'the social relationships among the device users must be taken into consideration during the design phase of IoT applications' [20]. Bao and Chen evaluate trust using both direct observations and indirect recommendations. The protocol is scaled to fit the needs of a smaller sub community of devices that share the same interests [20]. Trust is obtained through employing anomaly detection techniques, including high discrepancy in experienced recommendations, interval transmission, repetition, and delay rules [20]. Yet, in this model, trust is presumed based on social circles between devices' owners. This setup can be vulnerable to Sybil attacks. A trusted devices can be turned rogue after a period of trusted behaviour. Moreover, the cold start problem that occurs when new devices join a network, is not addressed.

To prevent the cold start problem in our model, we use a method proposed by Devi et al. [21]. Their work employs Probabilistic Neural Networks (PNNs) as a recommender system to calculate trust between users and predict ratings of newly added shopping items with zero ratings. Recommender systems 'search through large volumes of information to provide personalized content and services' [22]. For example, based on previous user behaviour choices and selections, a recommender system can predict and suggest possible options, such as movies or restaurants. A similar approach can be used in establishing trust between IoT devices. IoT is made of a group of connected devices, and nodes should be able to determine whether it is safe or not to exchange data with other nodes on the network. A recommender system can be used to decide on connection to a node, when it is safe to do so, based on previous observed behaviour. Various approaches to design recommender systems have been suggested in the literature: including *collaborative filtering*, *content based* and *hybrid approach*. Collaborative filtering finds similar users based on similar rating behaviour and provides recommendations accordingly [21]. This approach takes into consideration other users opinions' or actions'. Content-based recommender systems use attributes such as the description of items to recommend similar items. A hybrid recommender system combines both collaborative filtering and content based techniques. For trust prediction and calculation, we use a collaborative filtering technique, that we believe provides a good fit for IoT environments. A typical collaborative recommender system requires users and items. IoT devices can be treated as a group of users sharing common interests. Human users are profiled, and so are IoT devices, as in a way, they are considered the users of the system, machine users. Human users have characteristics, likes, dislikes, history and social circles. Devices have characteristics of their own as well and they can be profiled based on these characteristics. For example, transmission rates, number of successful packets delivered and number of dropped packets can be used. Our device trust calculation and evaluation is treated as a classification problem, hence, we use PNNs to address this problem. In this set up, recommendations from other devices (feedback) are weighted and these weights have to be updated and adjusted continuously. The network has the ability to learn behaviour over time and detect suspicious activities or transactions [23]. Our proposed model attempts to overcome limitations of other trust models by training the probability neural network and tuning its weights depending on devices' similarity and the nature of transmitted data. A probabilistic neural network is an implementation of the kernel discriminant analysis in which the operations are organized into a multilayered feedforward network with four layers: *input, pattern, summation* and *output* layers [24]. New data is used to modify decision boundaries in real-time and implementation can be through neurons that operate entirely in parallel [24]. This makes them efficient for IoT setup. The main advantage of PNNs is that they are constructed and trained fast, in one pass, unlike backpropagation algorithms that require multiple iterations to construct the model [21]. PNNs have a free parameter, which is the smoothing factor, and it can be adjusted without the need to retrain the network [25]. There are various levels to achieve parallelism for PNNS, including layer, node and bit parallelism [26]. We believe computing parallelism on the node level is the most fitting for the IoT and blockchain setting. Using this approach, we can solve the cold start problem for new devices that have just joined the network and obtain more flexibility in decision making. We discuss the details of trust calculation in Chapter 3.

Table 2.2 shows a summarizing comparison of the reviewed IoT trust models. Based on our research and review of the above models, we have deducted that an effective IoT trust model should possess all of the following properties:

- The ability to validate identities of IoT devices on the network.
- The ability to prevent replication of existing identities.
- The ability to prevent issuance of bogus new identities.
- The ability to detect malicious nodes (Sybils).
- The ability to evaluate trust of IoT devices on the network and constantly refine evaluation results.

- The ability to verify integrity of recommendations (feedback) made by IoT devices on the network.
- The ability to prevent modifications of previous recommendations for the purpose of influencing devices reputations.
- The ability to protect against common network attack, such as Replay Attacks.
- The ability to work in different IoT contexts.

Our proposed trust model uses permissioned blockchains to manage identities and evaluate trust of devices on the IoT network. Identities cannot be replicated or impersonated, which protects the network from Sybil attacks. Trust evaluation results are constantly refined with each transaction, allowing detection of malicious nodes on the network. An immune tamper-proof record is used to maintain transactions' integrity and prevent modification of previous device transactions and trust recommendations. Additionally, our model takes into consideration the limited resources existing on most IoT devices, as well as devices' heterogeneity, by assigning key management, identity verification, transaction validation and record keeping to trusted capable entities on the network. Currently, none of the existing IoT trust models use this approach. In Chapter 3 of this thesis, we show how our proposed model addressed all the shortcomings of the models reviewed above.

Table 2.2: Comparison of Existing IoT Trust Evaluation Models

Criteria	**TRM** [7]	**TMS** [1]	**DDTMP** [20]
Processing	Distributed	Centralized	Distributed
Deployment Environment	Sensors in Cyber Physical Systems (CPS) only	IoT	IoT
Trust Evaluation	Quality of Service (QoS) metrics to analyze nodes' behaviour	Direct observations and recommendations	Recommendations and behaviour.
Trust Computation	Through applying fuzzy logic to predict trustworthiness of nodes.	Through weighted average of combined recommendations	Anomaly detection through finding suspicious high discrepancies between given recommendations and previpus ones to detect malicious devices

2.3 Blockchains

Extensive work has been introduced in blockchains for management of digital cryptocurrencies such as Bitcoin[3]. Blockchains are the core technology behind these currencies. The distributed and append-only nature of blockchains improves security and integrity of financial transfers. Regardless of the type, permissionless or permissioned, basic concepts in blockchains remain unchanged. The difference between the two types is that the latter requires network devices to be identified, authenticated and enrolled by a central authority, preventing them from joining the blockchain network directly, unlike a permissionless blockchain. Going forward, we use the term *business network* to refer blockchain networks in different contexts. Blockchains work by logging all transactions that occur on a business network in a hash-based structure referred to as *block*. A transaction can refer to a financial transfer, change of ownership of an asset, such as cars or bonds or information exchange between two participants on the network. Each block of transactions contains the hash of the previous block. Merkle Hash Trees (MHT) are used to keep track of hashes and authenticate transactions [27]. The obtained ledger can then be stored in a database or as a flat file [28]. In this section we discuss blockchains and analyze the architecture of permissioned blockchains that are used in our proposed model.

2.3.1 Structure

A blockchain is built up of sequential blocks that can hold different types of transactions. Every block is linked to its predecessors by containing the hash of that block. Every hash uniquely identifies its block. The blockchain uses SHA256 hashing algorithm to generate block hashes. A visualization of blockchains structure is shown in Figure 2.2.

---

[3] Bitcoin is a cryptographic currency for a distributed payment system on the Internet. Blockchains are the core technology of Bitcoin.

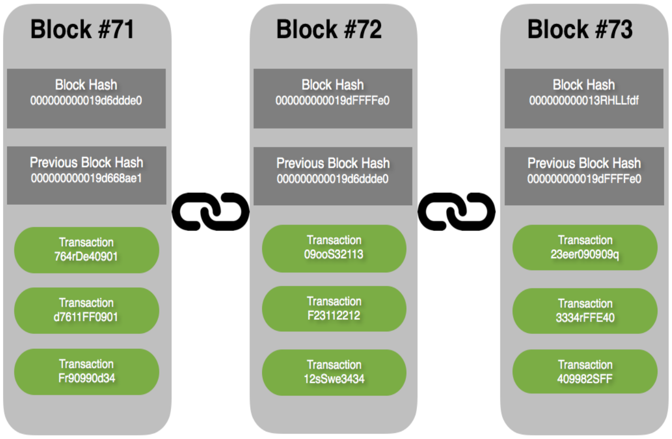


Figure 2.2: Blockchain Structure

2.3.2 Blocks

The series of block hashes referencing each block to its parent block generates a chain that traces back to the very first block, known as the genesis block [28]. Since a child block refers back to its parent block, the child block's identity is dependent on its parent. In other words, a change in the parent block hash reflects a change in the child block hash as well. 'This cascade effect ensures that once a block has many generations following it, it cannot be changed without forcing a recalculation of all subsequent blocks' [28]. This is one of the key features of blockchain's security. In order for an adversary to modify the sequence, they have to recalculate all the blocks tracing back to the genesis block. This process consumes a huge amount of computations, especially that the longest chain is always trusted. Therefore, the blockchain ledger is immutable [28].

Two main parts constitute a block. The block's header and transactions. In the header section of the block, the following information is stored:

- Time Stamp:

  Indicates the block's time of creation.

- Block Hash:

  Hash value of the current block.

- Previous Block Hash:

  Each block in the chain contains a SHA256 hash referencing back to the previous block.

- Transaction Root: A Merkle tree that summarizes all transactions in the block. MHT ensure blockchain's integrity. It facilitates detection of any changes made to a block's transactions. Merkle Trees are data structures represented by binary trees. Every block in the chain contains a number of transactions and the hash of each transaction is calculated. This is achieved through hashing the transactions through SHA256, twice, making up the leaf portion of the tree. The combination of two transactions is then hashed again in the same manner. This process is repeated until we end up with one hash at the root of the tree, known as the root hash. The chain runs through the hashes from top to bottom to verify no changes were made. Figure 2.3 shows an example of a MHT of four transactions.

2.3.3 Nodes

In permissioned blockchains, two main types of nodes exist to facilitate blockchain operations. Nodes that are responsible for validating, committing and ordering transactions into blocks are known as *validating peers*. Other nodes are known as *non-validating peers*, which can include network participants, such as IoT devices. Transactions are usually ordered into blocks based on a predefined batch size per block.

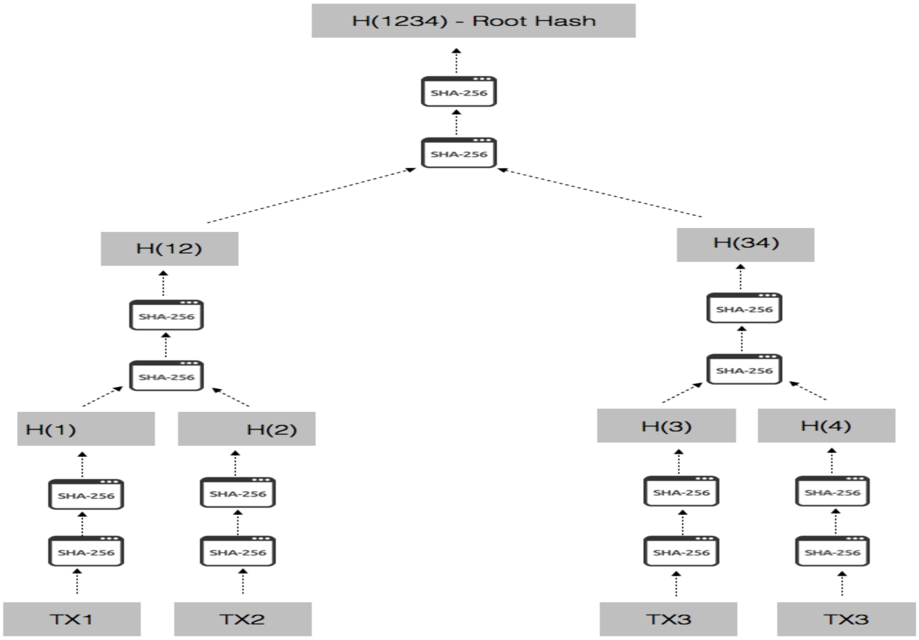


Figure 2.3: Example of Blockchain Merkle Hash Tree with Four Transactions.

2.3.4 Main Functionality

Our trust model uses permissioned blockchains to implement trust evaluation logic. Features of permissioned blockchains are listed below [29]:

- Identity Management

  Typical permissioned blockchains support membership services for identity management and authentication of nodes on the network. Nodes connected to the blockchain network know each other's identities.

- Concurrent processing

  Different types of nodes can exist on a permissioned blockchain network. These nodes are assigned roles based on their type. Some nodes are responsible for validating transactions on the network. Others are responsible for ordering transactions into blocks. In our trust model, this separation of responsibilities reduces processing overhead on IoT device that are connected to the blockchain network. Processing required for authentication and authorization is limited with this divergence in labour.

- Smart Contracts

  Permissioned blockchains can utilize smart contracts to implement the logic behind the blockcahin. It is used to control parameters passed during transactions, defining assets and their ownership. Such parameters can manage changes made to assets transfer of ownership under predefined rules.

2.3.5 Permissioned Blockchain Architecture

Typical architecture consists of different types of nodes, referred to as *peers*. Types of peers include: validating and non-validating peers. Validating peers are responsible for maintaining the ledger's integrity by validating transactions and updating the ledger [30].

Blockchain Components

A permissioned blockchain is typically composed of the following [29]:

- Network Participants:

  Participants in a blockchain network refer to end users that participate by submitting transactions on the network. For example, in our model, participants include IoT devices that interact with each other and need to verify trust.

- Chaincode:

  Chaincode is a piece of code that executes the logic of the blockchain through pre-defined smart contracts. A smart contract, similar to paper contracts, is a program that is executed if certain conditions and policies are met. Smart contracts have the ability to define assets, participants and instructions carried out by transactions to modify assets or participants' attributes on a blockchain network.

- Ledger:

  Distributed key-value database containing pseudo anonymous transactions between participants on the network. The ledger component comprises the sequenced list of all transactions in the blockchain network.

- Security and Membership Services:

  Permissioned membership ensures that transactions can be detected and traced by authorized regulators and auditors. All participants must have valid identities in order to submit transactions on the blockchain network. Generation of cryptographic certificates is performed by a Membership Service Provider (MSP) with CA capabilities. A permissioned blockchain CA can play different roles and is typically responsible for issuing three different types of certificates [29]:

  1. Enrollment Certificates:

     Enrollment Certificate Authority (ECA) registers new devices on the blockchain network and enables them to request a pair of enrollment certificates (ECerts). The first certificate is used to sign data and the second one is used to encrypt data. Embedded public key pairs implement Elliptic Curve Digital Signature Algorithm (ECDSA), in which data encryption key is converted by enrolled users.

  2. Transaction Certificates:

     Enrolled participants are able to request Transaction Certificates (TCerts) from a transaction certificate authority (TCA). Tcerts are used to deploy or invoke smart contracts on the blockchain network.

  To summarize, responsibilities of a CA include:

1. Registration of joining identities (new participants)
2. Issuing Enrollment Certificates (ECerts)
3. Issuing Transaction Certificates (TCerts)
4. Renewing certificates
5. Revoking certificates

2.3.6 Security in Blockchains

The majority of existing blockchain implementations employ PKI to secure communication between nodes and also authenticate transactions on the blockchain network. PKI frameworks provide distribution and identification of public encryption keys and digital signatures, which enable network participants to exchange messages securely. This assures messages' source and authenticity as well as confidentiality. Yet, a decentralized public ledger where participants witness anonymous transactions flowing on the network can be vulnerable to an attack known as the 51% attack. Typically occurring in permissionless (public) blockchains, upon gaining full control of the mining[4] capabilities, adversaries have the ability to control the process of creating new blocks if they control the majority of the processing power. This allows adversaries to selectively choose transactions within a block and consequently monopolize the blockchain network.

2.4 Blockchain and IoT

Blockchain solutions were originally revolved around cryptographic currencies. Hence, very little work has been explored in blockchains and IoT networks. Currently, one trust model

---

[4] Mining is a concept mainly related to Bitcoin. It refers to heavy cryptographic operations performed by nodes on the blockchain network to create new blocks that can be appended to the blockchain. It is a consensus mechanism to verify transactions.

exists that uses blockchains in P2P Wireless Sensor Networks (WSN). Moinet et al. [31] proposed a 'Blockchain based trust & authentication for decentralized sensor networks'. The authors use blockchains' architecture as a decentralized storage to store sensors cryptographic keys and trust information [31]. Their main focus is identity and trust management in WSNs. The model is designed for WSNs. According to the authors, WSN can be defined as decentralized networks that are composed of resource constrained nodes, such as embedded devices. Their model is represented as an undirected graph G = (V,E), where a node is described as a vertex and the edge represents a link between two nodes. In the model, a Network Node (NN) defines a vector of node properties and node abilities. Available Services (AS) defines dependencies of nodes abilities, such as their resources [31]. Every NN and SA is associated with cryptographic keys on the network. The authors base the key management component on Pretty Good Privacy (PGP) encryption standard, which is an implementation of PKI. A master key is used to identify a NN or an AS during network's connection time. In their work, *data payload* refers to a transaction or event that provides information about cryptographic keys or NN status. A node submits a *credential payload* that contains its master public key in order to be authenticated. As observed in permissionless blockchains, such as Bitcoin, nodes on the network race to solve a mathematical problem determined by the system to create new blocks. This process is known as *mining*. In mining, nodes attempt to solve a puzzle that allows them to obtain the hash of the new block, which is known to be computationally demanding. The miner, which is a node that performs mining, can select payloads in the block, since it is the one creating it. The authors state that only authenticated miners are allowed to mine new blocks. For a block to be recognized as valid, it must contain a Miner Approval (MA) payload that is generated by the miners. When a new device joins the network, it is issued a Credential Payload (CP), containing its public key. For an authentication request to be approved, the CP must be a part of a valid block. The status of a NN credential can be either renewed or revoked.

For trust management, a separate component is used to calculate trust of nodes on the network. This trust model is called 'Humanlike Knowledge based Trust (HKT)', which is based on human like behaviour for maintaining nodes' reputation on the network [31]. According to the authors, 'HKT is a compromise between a mutual surveillance by all nodes on the network and the presence of a trust center' [31]. HKT uses payloads stored in the blockchain to characterize nodes' behaviour to calculate trust levels. Every type of payload is mapped to an event that associates the payload to a reputation factor. After that, reputation levels of each NN is broadcast to the whole network to prevent the possibility of modification. At the end, trust in their model is expressed as a probability that an NN will perform a certain task correctly. Trust evaluation is performed through a comparison of current reputation level of a NN against a minimum reputation level defined by the model. With this model setup, malicious nodes that are authenticated can abuse the network by overloading the system with valid payloads. However, the authors address this issue by defining timing rules specific to submitting payloads. The rules enforce a limitation in time between payloads submitted by a NN.

In review of the above blockchain trust model, we identify the following shortcomings, which are addressed in our IoT trust model, and show how our work differs from the reviewed model:

1. The authors state that once a CN is revoked, the NN is issued a new one in order to stay authenticated on the network. This does not solve the internally malicious node problem. A node with revoked credentials should be examined or disconnected from the network if it is compromised. It should not be allowed to re-join the network by requesting a new CP.

2. The model employs the architecture of a permissionless blockchain. This allows any device, malicious or benign, to join the network. Even if a malicious node joins the network and the trust model component detects it at a later time, the fact that the malicious node remained undetected for some time is unchanged. During that time, the malicious node can carry out its malicious tasks.

3. Mining is a term specific to digital currencies, Bitcoin in particular. The reason behind the naming is that miners receive a reward (mining fee) for their mining efforts, which is a small amount of Bitcoins. The term used in this model is a bit misleading, as there is no coins to mine. Additionally, the proposed model suggests mining to take place on wireless sensors. We believe that mining on IoT devices and sensors can be computationally infeasible due to the limited resources on the majority of these devices. Mining on devices indicate that they will store blocks as well. A permissioned IoT blockchain eliminates that by distributing authentication, transaction validation and block ordering to separate capable entities on the blockchain network. As a result, processing required on IoT devices' side is minimized to submitting HTTP requests over a RESTful API. Additionally, we can enforce a more trusted environment for IoT devices through applying smart contracts to further evaluate trust among IoT devices and guarantee that only valid devices with trusted identities are permitted to submit transaction on the network.

4. A scenario where a miner node is compromised is not addressed. Nevertheless, this model is vulnerable to the 51% attack, discussed in the previous section. It is possible for an adversary to control the majority of the miners on the network. This allows adversaries to selectively choose which payloads to include in a block, and as a result, maliciously affect reputations of certain nodes on the network.

5. Instead of separating the trust management component from the authentication and

key management component, our model allows all of these components to work together as part of one system.

6. Our model defines clear access control rules that guarantees transactions can be submitted only by authorized devices.

A key aim of our proposed model is to improve security of IoT networks by protecting them from several common attacks that were discussed in Section 1, including Sybil attacks. A secure IoT trust model would be able prevent both external and internal Sybils from disrupting the network. While the model reviewed above proposes a similar identity management approach, the methodology of trust evaluation is completely different. Our model utilizes smart contracts to dynamically evaluate trust of participating devices and refines that trust immediately after a transaction. This is achieved through rewarding a device with high ratings for its trusted service or punishing it with low ratings. This facilitates detection of internal Sybils, in the case of node compromise. Moreover, the integrity of reward and punishment transactions are protected with an immune history to prevent adversaries from modifying previous device ratings. Additionally, our model enrolls IoT devices from a trusted CA to manage devices identities throughout the network's life. All devices on the blockchain network must be issued enrollment certificates from the blockchain CA. Those certificates act as participants' proof of identity documents on the network to protect their identities while communicating across the network.

2.5 Conclusions

In this chapter, we discussed IoT's architecture, its security requirements and challenges. Moreover, we explored the concepts of trust to introduce IoT trust models. A review of some of the existing IoT trust models related to our work was covered as well. Additionally, we discussed common attacks on trust models and IoT networks. Finally, we explored the components, architecture and security of blockchains.

Chapter 3

Proposed Model

In blockchains, connected nodes have the ability to see neighbouring nodes' transactions and hence, compromised nodes across the blockchain network can be detected. Additionally, multiple synced copies of ledgers containing all transactions are stored on multiple nodes on the blockchain network. All copies of the same ledger must be in sync. Therefore, attempts to make changes to a single copy will be contradicted by the whole network. Applying this distributed approach to IoT networks can improve IoT transactions' integrity and confidentiality. In this chapter, we discuss the details of our proposed model, its phases and how it can be used to verify IoT devices' identities and detect malicious activities across the IoT network. Additionally, we discuss implementation details of a proof-of-concept prototype that implements some of the key components of our proposed trust model. We use Hyperlegder Fabric tools to implement our prototype.

This chapter is organized as follows. Section 1 provides an overview of our IoT trust model. Hyperledger blockchains are reviewed in Section 2. Section 3 details the trust model's specifications and its phases. In Section 4, we discuss our prototype implementation. We discuss obtained implementation results in Section 5. Section 6 concludes this Chapter.

3.1 Model Overview

IoT networks can contain various devices diverging in resources and capabilities. Some devices are blessed with more computational power and memory, while others are not. Utilizing these available resources efficiently is key for conserving energy of less capable nodes. From that vector, our proposed model takes IoT's heterogeneity and distribution into consideration. The trust model introduced in this thesis can be considered decentralized. A level of decentralization must be allowed, as this model is of a permissioned (private) blockchain that can be implemented in parallel with an existing IoT network. Permissioned blockchains allow for utilizing different nodes on the blockchain network to carry out different tasks and responsibilities. In addition, participating entities are enforced to be enrolled by a CA on the blockchain network. Multiple CAs can exist on the same blockchain network to distribute devices' enrollments into the network. However, our implementation provides a proof-of-concept only; therefore, one CA exists in our current model. In our model, transactions are verified in order to be appended to the blockchain ledger. This responsibility is taken by multiple *peer* nodes on the network. Peer nodes are not to be mistaken with IoT devices. We refer to IoT devices as network *participants* or *clients*, the two terms are used interchangeably. Peer nodes on the other hand are trusted and dedicated blockchain facilitators that are. Certain peer nodes can be assigned processing of certain IoT devices on the blockchain network.

Our primary objective is to improve security of IoT networks by introducing an effective trust model that can be resilient to the common attacks discussed in Chapter 2. Mainly, a model to protect against Sybil, bad or good mouthing and wormwhole attacks. Effective trust and reputation models should detect malicious behaviour to help IoT devices determine which nodes are indeed safe to communicate with and which nodes are malicious. Our goal is to measure trust of IoT devices and enforce authentication, local authorization and integrity of past and future transactions on the network. To lay out our model, we list its key components below:

- Identity Management:

  All participants on the blockchain network must be issued *enrollment certificates* from the blockchain CA. Those certificates act as participants' proof of identity documents. Identities are issues, renewed and revoked only by a CA. This allows only for authorized devices to join the IoT network. Additionally, connected devices must be authenticated ahead of joining the network. As a result, participants are able to cryptographically sign submitted transactions on the blockchain network [32].

- Trust Evaluation:

  Trust evaluation and refinement is an important component of our proposed model. We treat trust evaluation as a classification problem and we use PNNs to address this problem. In this set up, recommendations (devices' feedback) are weighted and these weights have to be updated and adjusted continuously. The model has the ability to learn behaviour over time and detect malicious nodes that do not meet trust requirements. Our proposed model attempts to overcome limitations of other trust models by training the probability neural network and tuning its weights depending on devices' similarity and the nature of data that is transmitted. We use PNNs to calculate and evaluate trust between IoT devices and predict trust scores for new devices as well.

  Trust calculations for each device are based on its behaviour history and recommendations given by other devices on the IoT network. Recommendations are expressed in terms of *trust points* and translate into *trust scores* after receiving a service from an IoT device with either reward or punishment. IoT devices' recommendations (possible trust points) range between 0-5, where zero indicates poor service and fivefffx indicates an excellent one. Using PNNs, devices can be classified to trustworthy or untrustworthy. Given trust points show the rating of the service received by the device.

Distributed processing is performed by multiple blockchain *peers* that execute *smart contracts*. Smart contracts manage and determine the logic behind the blockchain's transactions. Invocation of defined functions results in a transaction on the blockchain network. Trust calculation and evaluation are performed through invocation of smart contracts functions that perform PNN classification. Devices are deemed trustworthy if approved by multiple peers. Each IoT device bears a rating that helps determine its trust score. Trust scores are updated by IoT device directly after interaction or data exchange activities between two IoT devices. The requesting IoT device *rates* the interaction from the serving IoT device by either rewarding it with high trust points or punishing it with low trust points.

- Local Authorization:

  In our proposed model, we refer to the stronger nodes in the IoT network as *Alpha nodes*. This is because *Alpha nodes* can play the role of validating peers in the blockchain network. Additionally, since our model is decentralized, *Alpha nodes* can be responsible for maintaining administrative functions of the blockchain network to better manage connected IoT devices. For these nodes, we assume the following:

  1. Alpha nodes are more capable nodes that are configured once at the model's setup.
  2. Alpha nodes are most likely IoT control hubs that are connected to a power source or have long life batteries.
  3. Membership of Alpha nodes in the network does not frequently change.
  4. Alpha nodes have administrative privileges.

5. Alpha nodes have the ability to represent edge nodes that can act as peer nodes capable of executing chaincode if needed.

Multiple alpha nodes can exist and their work starts at the first phase, data collection and profiling. Phases of the proposed model are described in the next section.

- Transaction Integrity:

  The shared replicated ledger component implements the blockchain technology. All transactions taking place on the network are written to a key-value database accessible with read-only permissions for all participants across the blockchain network to see. In order for an adversary to modify a single transaction, they have to recalculate all hashes of all previous blocks tracing back to the first block. This prevents adversaries from manipulating previous trust recommendations and devices' trust score.

- Data Sensitivity:

  In IoT security, the flow of information should be taken into account, as all sorts of it fly across the network. Some information is made available in public, such as weather readings or bus locations. However, some information should be confidential, such as patients' information. Therefore, different levels of security should be implemented depending on the information transmitted. This is recorded once the device is connected and fed to the model as an input parameter. For simplicity, the model interprets this as a flag parameter of the device. If the flag is set to 0, then the information is publicly available and the model will depend on the available trust score of the serving device to determine its' trustworthiness, based on a predefined threshold. If the flag is set to 1, then the information is confidential and the model will perform trust computations.

- Strictness Level:

  The strictness of trust evaluation can be tuned to adjust to different settings. A threshold pre-defined system parameter is used to adjust the network's strictness during trust evaluation. The model allows for three levels of difficulty:

  - Easy: When *easy* difficulty is set, the model evaluates devices' trust scores against a minimum threshold. Only transactions submitted by devices with trust scores exceeding that threshold value are allowed.
  - Medium : A more strict setting with a threshold set to 50% of trust score's maximum value.
  - Strict: An extremely strict model evaluation setup that allows devices to hold at least 70% of trust score's maximum value.

3.2 Hyperledger Blockchain

In order to meet our IoT trust model goals, we needed a blockchain tool that can support distributed file sharing and dynamic task coordination. Therefore, we used open-source tools from the Hyperledger Blockchain Project. Currently, three open-source frameworks exists for blockchain development:

- Hyperledger Fabric:

  Hyperledger fabric is an implementation of permissioned blockchains for developing blockchain applications. Fabric utilizes chaincode in Go and Java languages to leverage smart contracts, which execute blockchain logic.

- Hyperledger Iroha:

  Hyperledger Iroha is another blockchain development platform that is developed in C++ and puts emphasis on mobile application development.

- Hyperledger Sawtooth Lake:

  The Sawtooth Lake model is driven towards modularity and financial use cases. It utilizes a consensus algorithm known as Proof of Elapsed Time (PoET).

For our model implementation, we chose to use the Fabric model because of documentation availability. Additionally, Fabric is the only Hyperledger implementation that supports integration with logic modeling tools such as Fabric Composer [1].

Hyperledger Fabric utilizes the use of *peers*. Validating peers are responsible for maintaining the ledger's integrity by validating transactions and updating the ledger [29]. Fabric model architecture is illustrated in Figure 3.1.

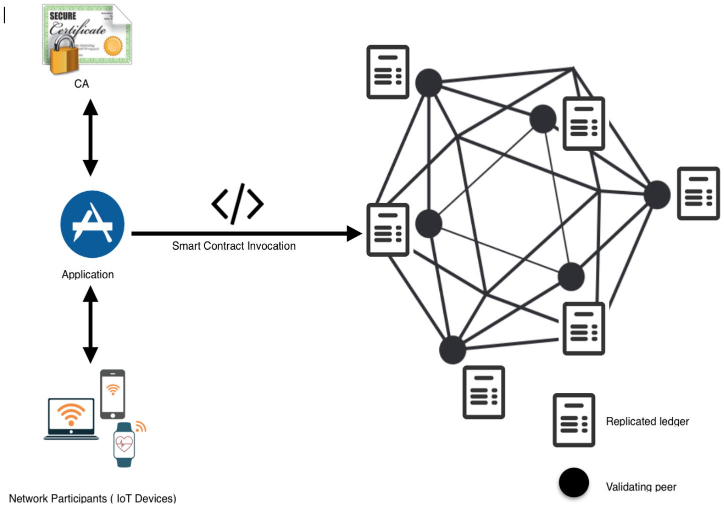


Figure 3.1: Hyperledger Fabric Model Architecture

3.2.1 Blockchain State and Transaction Flow

Before discussing the detailed phases of our model, we discuss how transactions are processed in the blockchain tool used in our model's prototype.

---

[1] Fabric Composer is a tool for defining and modeling blockchain business networks: https://hyperledger.github.io/composer/

Blockchain State

In the Hyperledger Fabric model, a blockchain state that reflects the latest transactions is designed to be saved and version-labeled as a key-value store (KVS). In KVSs, keys represent names and values for every transaction. *Chaincode* software manipulates these entries that run on the blockchain through two KVS operations, *put* and *get*. The state of the blockchain is persistently stored with logged updates.

Transaction Endorsement and Workflow

Different blockchain implementations handle transactions differently. In Fabric, transactions are proposed, simulated and validated before being committed to the blockchain network. We discuss the high-level flow for a transaction in the Fabric blockchain network in the form of a request initiated from an IoT device participant.

Endorsement Policies:

According to Hyperledger Fabric documentation [2], an endorsement policy is defined as a condition that determines a transaction's endorsement or its rejection by *endorsing peers*. An endorsing peer role is to verify the following about a proposed transaction:

1. The transaction is properly formed and contains all required parameters.
2. The transaction has not been submitted previously. This is to prevent against message replay attacks.
3. The transaction is carrying a valid client signature. This is done using an MSP component (CA).
4. The client submitting the transaction is authorized to perform the proposed operation.

---

[2] https://hyperledger-fabric.readthedocs.io

Endorsement policies cannot be modified, as they are deployed at chaincode instantiation at blockchain creation, which occurs once when setting up the network. This is to help prevent dynamic addition or deployment of new policies. If the proposed transaction is compliant with the defined polices, after endorsing peers' assessment, it is declared valid; otherwise, dropped. Figure 3.2 illustrates a typical transaction flow. Blockchain transaction flow is explained below for a transaction initiated by a smart refrigerator requesting room temperature.

1. A transaction is initiated by a client (a smart refrigerator) to request current room temperature. This request results in a PROPOSE message that is sent to a predefined group of endorsing peers on the blockchain network.

   The format of a PROPOSE is $< PROPOSE, tx >$. Based on predefined endorsement policies, some peers might choose to object or endorse the proposed transaction. Since endorsement is related to defined chaincode, we evaluate participants trust scores to allow or prevent the transaction proposed. Trust in granted only if a defined number of peers endorse the transaction. In other words, the IoT device that initiated the transaction must collect enough *endorsements* in order to be approved. Transaction parameters include:
   $tx =< clientID, chaincodeID, txPayload, timestamp, clientSig >$ where :

   - clientID represents the ID of the client submitting the PROPOSE message.
   - chaincodeID represents the chaincode pertained by the transaction.
   - txPayload represents the payload that contains the transaction data.
   - timestamp represents an integer number maintained by the client which is monotonically increased for every new transaction.
   - clientSig contains the signature of the client.

Additionally, a unique cryptographic hash is used to identify the transaction tx. In other words, $tid = HASH(tx)$. Once the request is submitted, the client awaits for the endorsing peers' response.

2. Upon receiving the PROPOSE message, endorsing peers simulate and execute the transaction using transaction arguments to invoke chaincode. A signature is produced for that transaction. This occurs after the client's signature (clientSig) is verified by the MSP. Until this point, no updates are made to the blockchain ledger.

3. Produced endorsement is then collected and broadcast by the submitting client. Once a predefined number of endorsement messages is received by the submitting client, indicating approval and endorsement of the proposed transaction, an ordering service is invoked to commit the transaction and add it to a block. If the client does not possess the ability to invoke an ordering service, it can convey the broadcast through a trusted peer. Even in that case, an endorsement still cannot be fabricated, as it will be verified as well.

3.3 IoT Trust Model

We propose a model design that integrates blockchain's architecture with a reward and punish mechanism to ensure trust in the IoT network. Figure 3.3 shows our model's flowchart. In this section we introduce the phases of our proposed IoT trust model.

3.3.1 Model Phases

The general modeling process and phases are depicted in Figure 3.4.

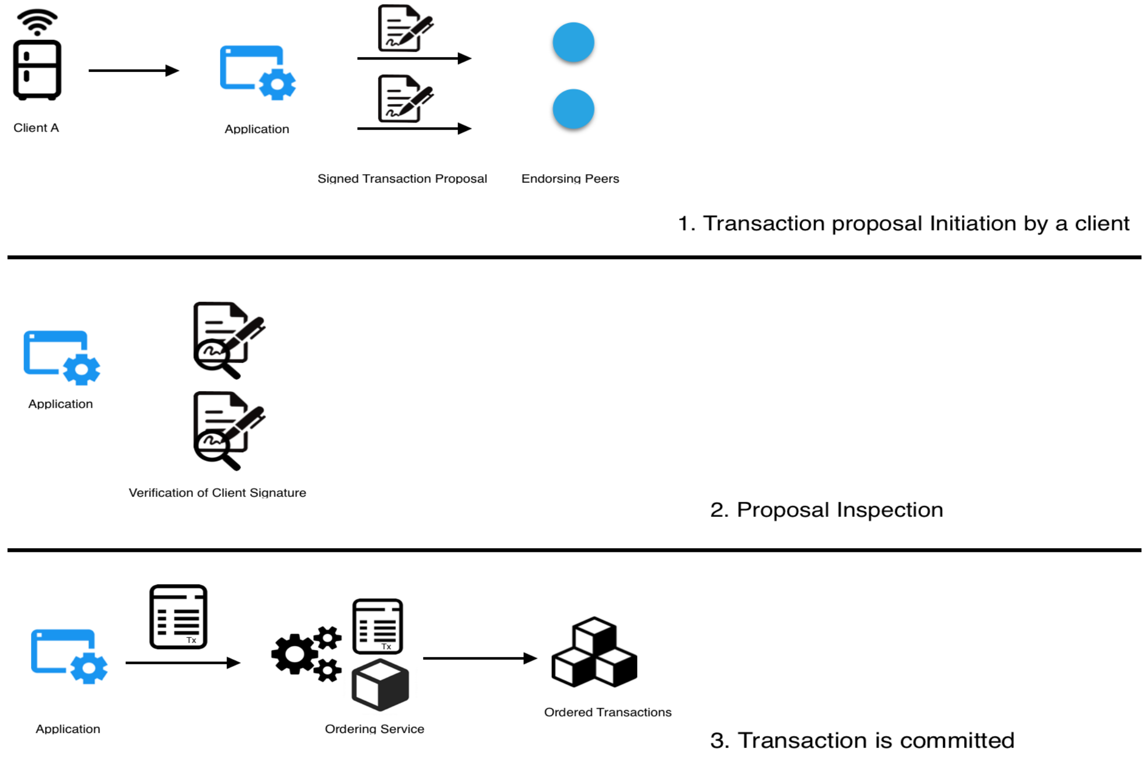

Client A
Application
Signed Transaction Proposal
Endorsing Peers
1. Transaction proposal Initiation by a client
Application
Verification of Client Signature
2. Proposal Inspection
Application
Tx
Tx
Ordering Service
Ordered Transactions
3. Transaction is committed

Figure 3.2: Transaction Flow

Information Collection

IoT devices are constantly listening, collecting and transmitting data. In our model, alpha nodes and blockchain peers are constantly listening and storing the following information about nodes: transmission rate, successful packets ratio, dropped packets ratio, battery life, CPU power, available memory, trust score and severity flag. IoT devices are required to provide feedback for their experience upon completion of a transaction, similar to a service rating. These ratings are associated with a timestamp and taken into account later in trust assessment. To guarantee rapid communication, each IoT device maintains a *trust score* that is stored in the blockchain for other IoT devices to see. We mentioned in our model overview that we capture a severity flag that specifies the nature of information being transmitted. If the flag is set to 0, then the information is publicly available and the blockchain peer nodes will only evaluate the device's existing trust score against a predefined threshold to determine

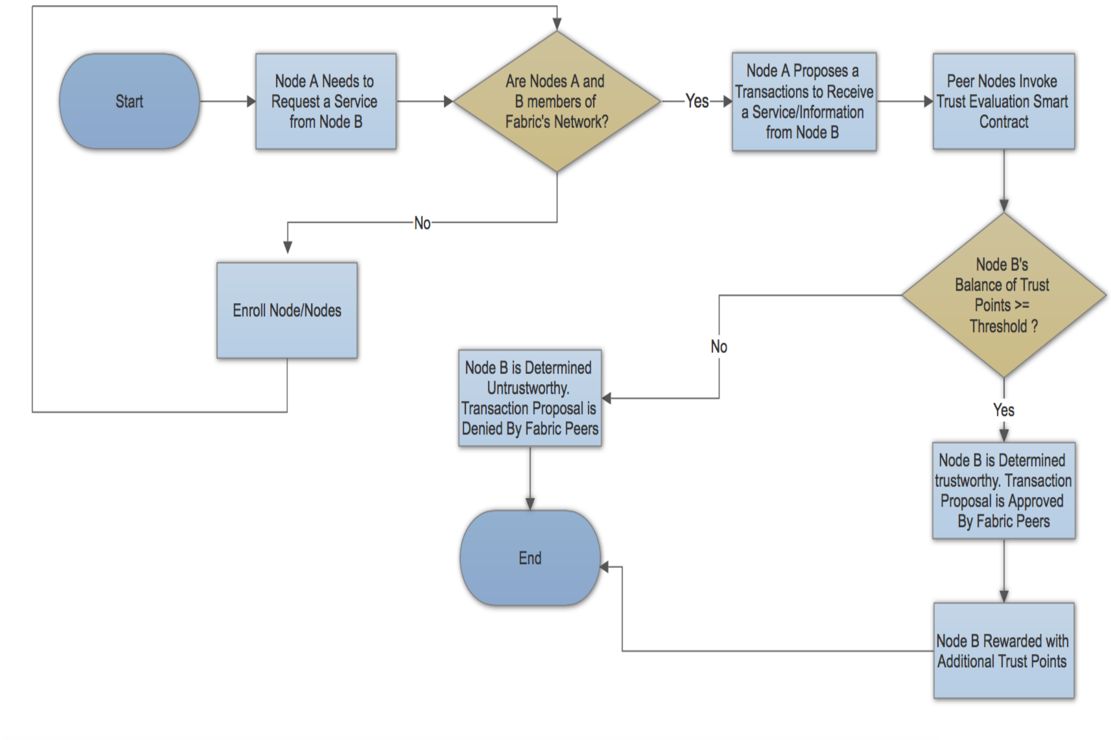

Start
Node A Needs to Request a Service from Node B
Are Nodes A and B members of Fabric's Network?
Yes
No
Enroll Node/Nodes
Node A Proposes a Transactions to Receive a Service/Information from Node B
Peer Nodes Invoke Trust Evaluation Smart Contract
Node B's Balance of Trust Points >= Threshold ?
No
Yes
Node B is Determined Untrustworthy. Transaction Proposal is Denied By Fabric Peers
Node B is Determined trustworthy. Transaction Proposal is Approved By Fabric Peers
Node B Rewarded with Additional Trust Points
End

Figure 3.3: Proposed Model's Flowchart

trust. If the flag is set to 1, then the information is confidential and peer nodes will continue trust computations and device classification.

Node Enrollment

At the start of the model's setup, bootstrapping of devices that join the network is performed. The CA provides keys and authorizes IDs of devices joining through MSPs. This is done by registration and enrollment of a new user (the new IoT device). The administrator managing the blockchain registers a new user with Enroll ID. The device then enrolls and receives credentials from the CA. Next, the device requests Tcerts from the Transaction CA in order to submit transactions. An adversary cannot force bad ratings by punishing a certain devices with negative feedback to deny it service or to ruin its reputation on the IoT network. Additionally, forcing IoT devices to interact with valid identities only, allows for the ability

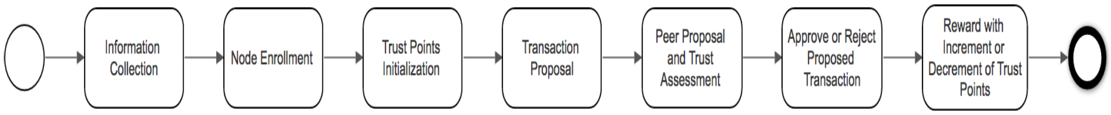

Information Collection
Node Enrollment
Trust Points Initialization
Transaction Proposal
Peer Proposal and Trust Assessment
Approve or Reject Proposed Transaction
Reward with Increment or Decrement of Trust Points

Figure 3.4: Modeling Process

to verify the source of submitted transactions.

To determine which peers are assigned for the enrolled IoT device, collected information about the device is used. Devices' model, capability, available resources and information level are stored at the beginning. All these features are used in the PNN to determine similar devices and assign them to certain peers accordingly. Every blockchain peer can be responsible for a certain category of IoT devices. To calculate similarity, we assume there are N client devices and M server devices. Client devices are devices that require a service, while server devices are devices that should provide a service and are suspected to be untrustworthy. At the beginning of the model's life, artificial transactions are forced to create interactions between nodes, as the network needs data to learn. Most devices will start with a rating of zero and that is resolved by predicting a rating (trust score) value for those devices that have just joined the network. The rating for a service requested by node i and fulfilled by node j is represented by R_{ij} where $1 \leq i \leq N$ and $1 \leq j \leq M$. The smoothing factor σ is calculated. To obtain it, we first calculate the Euclidean distance between nodes i and j.

$$d_{ij} = \sqrt{\sum_{k=1}^{M} \mid R_{ik} - R_{jk}|} \tag{3.1}$$

Now σ is calculated:

$$\sigma_i = \mathrm{g} * \frac{\sum_j^N d_{ij}}{N} \tag{3.2}$$

In equation (3.2), g is a constant tuned at the training phase. N is the number of nodes and d is the calculated distance obtained in equation (3.1).

We assign same-category and similar IoT devices to a certain peer to find recommendation weights that contribute to devices' classification. Recommendations that are collected from devices in the same category and assigned to the same blockchain peer weigh more as we assume they are more important to the transaction. Quality of Recommendation (QR) is the score of trustworthiness that is assigned to a witness device depending on the accuracy of its past ratings [1]. This helps the model in protecting against bad and good mouthing attacks. QR's value ranges between [1,-1], with 1 representing a trustworthy witness node and -1 the opposite. The model gives more weight to devices belonging to the same category; in other words, based on the similarity between the two nodes, d_{ij}. In the weight calculation, equation (3.3), λ and θ are parameters ranging from [0, 1] and expressing the memory of the system. $(t_{now} - t_j)$ is the difference between the current system time and time the transaction took place; older ratings are less likely to be as important as new ones.

$$w_{R_{ij}} = \lambda^{d_{ij}} \theta^{(t_{now} - t_j)} \tag{3.3}$$

Trust Points Initialization

IoT devices that are part of the blockchain network maintain a *trust score* attribute that reflects their available balance of trust points and reflects their rating among other IoT devices. Upon obtaining an enrollment certificate and becoming part of the network, identity of the IoT device is recorded and cannot be forged. After successful enrollment, trust is computed immediately. Trust between nodes i and j is computed with equation (3.4), where R is the rating for the service requested by node i, fulfilled by node j, and w is the weight of the device. :

$$T_{ij} \;=\; \exp[\frac{1}{\sigma_i} * \{(R_i' * R_j) - \frac{1}{2} * (\|R_i * w_i * QR_i\|^2 \pm \|R_j * w_j * QR_j\|^2)\}] \quad (3.4)$$

Refining these results is important in order to minimize risks of bad or good mouthing attacks. The model handles that by calculating trust for indirect devices. The same calculations are used, except that input size varies. It is changed to NXN instead. This is to find trust for indirect neighbouring node requiring multiple hops. This is repeated until trust updating is less than the threshold value (δ).

Predicted rating for node i (client device) towards node j (server device) is smoothened after transactions. Whether the node has previous ratings or is new to the network, ratings are predicted as below:

$$P_{i,j} = \begin{cases} \frac{\sum_{u=1,u\neq i}^{N} T_{u,j} * R_{u,j}}{\sum_{u=1,u\neq i}^{N} T_{u,j}} \end{cases} \quad (3.5)$$

In equation (3.5), P denotes the predicted rating of node j to node i who requested the service. $T_{u,j}$ is the trust value that node i has in node j. R is the original rating value of node j. $\bar{R}$ is the average original rating of node j. Equation (3.5) is used when n(j) $>$ 0, where n is the number of original ratings of node j. If it is a cold start problem and no rating is available, we use equation (3.6):

$$P_{i,j} = \bar{R}_i \quad (3.6)$$

The cold start problem for newly joined devices, where no rating is available for them, is tackled by filling the average rating of devices. In equations (3.7) and (3.8), smoothened rating S for the two nodes is calculated for different scenarios.

$$S_{i,j} = R_{i,j} \text{ if } R_{i,j} > 0 \text{ (original rating)} \tag{3.7}$$

$$S_{i,j} = P_{i,j} \quad \text{otherwise (predicted rating)} \tag{3.8}$$

Where $S_{i,j}$ is the smoothened rating, $R_{i,j}$ is the original rating and $P_{i,j}$ is the predicted rating.

Trust Evaluation

The trust evaluation component is performed during the client's attempt to obtain endorsements once they initiate a *transaction proposal*. Before a transaction is fully committed to the ledger, a transaction proposal must be sent by the client to existing endorsing peers. The endorsing peers will execute chaincode and evaluate the trust rating (trust score) of a device against a predefined endorsement policy. The endorsement policy defines the system threshold. To evaluate trust worthiness, peer nodes perform device classification. Ratings of the devices and trust scores are computed, updated, and used, along with the history of the device's behaviour to classify the device in question as trustworthy or untrustworthy by choosing the maximum likelihood that it is worthy of trust or not. The summation layers of PNNs calculate the Probability Density Function (PDF) that calculates the likelihood that node i belongs to the untrustworthy or trustworthy classes. Summation layer obtains this likelihood by 'summarizing and averaging the output of all neurons that belong to the same class' [33]. Finally, maximum likelihood is chosen and with that, the device in question is classified. If classification yields the same probability for both classes, classification of the device according to the Bayesian decision rule is performed, which takes into consideration all previous summation layer neurons [33]. Equation (3.9), is used to calculate the probability that device x belongs to class i.

$$p_i(x) = \frac{1}{(2\prod)^{\frac{d}{2}}\sigma^d}\frac{1}{N_i}\sum_{j=1}^{N_i}\exp\left[-\frac{(x-x_{ij})^T(x-x_{ij})}{2\sigma^2}\right] \tag{3.9}$$

In equation (3.9), N_i denotes the number of samples in class i (trustworthy). Dimension of the pattern vector of the node is denoted by d, x_{ij} is the neuron vector [33]. Maximum probability is selected and the device is classified. If enough peers classify a device trustworthy, the transaction proposal on the blockchain is approved. This allows the client (IoT device) to collect enough endorsements, causing the transaction to be endorsed and the device to be deemed trusted; otherwise, the transaction is dropped.

Transaction

Once a device proposes a transaction to the network, the endorsing peers will verify the device's signature and inspect payload of the transaction to simulate it, ahead of committing it to the blockchain ledger. During this step, the chaincode deployed by the endorsing peers performs trust evaluation as pointed in the previous step. The transaction is carried out and committed to the ledger if it was approved by a predefined number of endorsing peers.

Balance Update

Requesting devices reward or punish the server device by rating their experience with trust points from 0-5 upon completion of a transaction to update devices' trust scores. Point 0 indicates a punishment for a bad transaction and 5 indicates complete satisfaction and reward. Committing peers are required to update the submitting client's trust points. The committing peers perform this task because they are responsible for committing all transactions as well as they have the capability to store and execute chaincode. Unlike the reviewed TRM model [7], trust of IoT devices in our proposed model is constantly evaluated and refined with each transaction. This is to address the possibility that a device can turn malicious after a period of trusted behaviour.

3.3.2 Protection Against Trust Models' Attacks

Common attacks that can be launched on IoT trust models include Sybil, reply, bad mouthing, good mouthing and ballot stuffing attacks. Sybil attacks were discussed in Chapter 2. Bad mouthing attacks occur by manipulating ratings for nodes. An adversary might force bad ratings for certain nodes to deny them service or to ruin their reputation within the community. On the other hand, good mouthing attacks aim at providing good ratings for malicious nodes to make them appear trustworthy. Finally, ballot stuffing attacks. Which occur when a malicious node attempts to increase the trust value of its malicious peers by providing dishonest ratings and behaving well at the same time [1].

Every IoT device holds a blockchain identity that is verified ahead of transaction execution. In addition to verifying that a submitted transaction carries a valid IoT device signature, it is evaluated for proper format. Additionally, submission timestamp is checked to ensure the transaction has not been submitted before. This protects our model from being affected by message reply attacks. Through this validation, Sybil nodes cannot influence the network, as they cannot replicate this type of identity. If the transaction passes these checks, the blockchain uses peer nodes to determine whether to approve or reject transactions. This is determined during trust evaluation invoked by the defined smart contracts (chaincode). Endorsing peers can detect if the same transaction has been submitted previously, which also helps in protecting against reply attacks.

Our model can protect against these attacks because it learns the characteristics of a trustworthy node and those of a malicious one. In addition, it helps that ratings' weights are updated constantly and history of behaviour and quality of these ratings are taken into account in calculations. Additionally, maintaining different levels of security based on the nature of transmitted information helps the model eliminate unnecessary processing.

3.4 Model Implementation

The complete design of our proposed model was discussed in the previous section. For implementation, we built a proof-of-concept prototype that implements a portion of the proposed model. Referring back to Section 1 of this chapter, we implement four of the discussed key component: identity management, local authorization, transaction integrity and strictness level. In our implementation, we refer to IoT devices as *participants*. For prototyping purposes, we make some assumptions to simulate trust evaluation in order to integrate it with the mentioned implemented components:

1. Participants maintain a balance of trust points, referred to as *trust score*. Trust scores range between 0-100, with 0 being least trusted and 100 indicating a trustworthy IoT device. The balance is updated by participants directly after interaction or data exchange activities between two participants. The requesting participant rates the interaction from the serving participant by either rewarding it with additional trust points or punishing it by points subtraction.

2. The amount of points to be given or taken away is determined by the serving participant's packet forwarding behaviour. We assume for implementation that a healthy packet forwarding behaviour indicates a trusted device. Since most IoT devices usually communicate wirelessly, they are expected to behave differently in the case of compromise by adversaries. This makes it possible for adversaries to control compromised devices remotely to disrupt the network and affect data generated by devices. Therefore, we chose to use devices' Packet Delivery Rate (PDR) that is captured from the network.

We use PDR in the prototype as one possible indication of trust. However, the prototype can work with different attributes for trust indication as system designers see fit.

3. Ahead of communication between two participants, the requesting participant evaluates the trust score of the serving participant and only proceeds with the transaction if the score is equal to or higher than a pre-defined system parameter (threshold). Insufficient balance of trust points results in transaction rejection.
4. We assume the network captures and updates PDR values for connected IoT devices automatically. The values are included as a device attribute. For prototyping purposes, these values are currently updated manually.
5. Every device that joins the network is initialized with a balance of 50 trust points. This balance of trust points constitutes the trust score of that device. For simplicity, trust points are forced to range between 0-100.

We have developed a proof-of-concept prototype locally on a machine running Mac OSX El Capitan. Processor and memory specifications are 2.6 GHz Intel Core i5 and 8GB of RAM. The implementation process of the model's prototype was broken down to three major phases:

1. Infrastructure and runtime setup
2. Blockchain Network Modeling
3. Integration

3.4.1 Infrastructure Setup

To build the infrastructure of the blockchain runtime component, we used Docker[3] images of Hyperledger Fabric. Hyperledger fabric leverages Docker containers to manage peers and their chaincode on the blockchain network. In other words, we can run multiple containers on the same host. The following containers were created:

- **Peer**:

  One peer node is running on the blockchain network. The peer node is responsible for execution and invocation of the network's logic with instantiated chaincode. The logic is defined in the network modeling phase.

- **Certificate Authority**:

  The CA container acts as the network's MSP. The fabric CA is responsible for devices enrollment into the network. All Cryptographic key distribution and identities management is performed by the CA.

- **Orderer**:

  Orderer nodes are part of an *Ordering service*. The ordering service maintains communication channels that clients and peers connect to. It offers a broadcast service for messages containing transactions on the blockchain network.

- **Database**

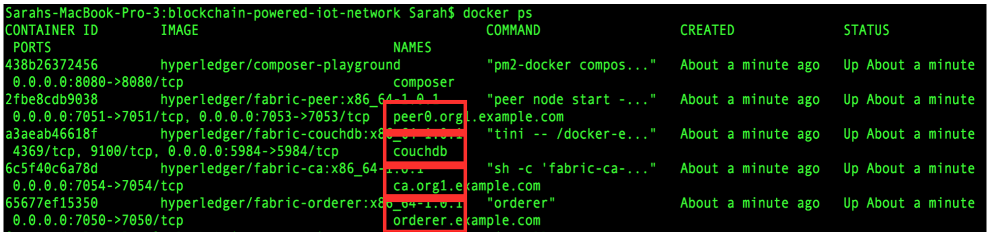
  The state database used is Apache CouchDB [4]. CouchDB is different from traditional relational database solutions, as it simply stores documents in key-value format. Additionally, it is JSON format compatible to match blockchain's data storage and chaincode queries as well.

---

[3] Docker is a container platform that allows for packaging code in a format where it can be run in an isolated setup within a shared operating system environment (https://www.docker.com/what-docker)

[4] http://docs.couchdb.org/en/2.1.0/intro/index.html

A captured screen listing the running blockchain container instances is shown in figure 3.5.



```
Sarahs-MacBook-Pro-3:blockchain-powered-iot-network Sarah$ docker ps
CONTAINER ID        IMAGE                                COMMAND                  CREATED              STATUS
 PORTS                                           NAMES
438b26372456        hyperledger/composer-playground      "pm2-docker compos..."   About a minute ago   Up About a minute
 0.0.0.0:8080->8080/tcp                          composer
2fbe8cdb9038        hyperledger/fabric-peer:x86_64-1.0.1     "peer node start -..."   About a minute ago   Up About a minute
 0.0.0.0:7051->7051/tcp, 0.0.0.0:7053->7053/tcp   peer0.org1.example.com
a3aeab46618f        hyperledger/fabric-couchdb:x86_64-1.0.1  "tini -- /docker-e..."   About a minute ago   Up About a minute
 4369/tcp, 9100/tcp, 0.0.0.0:5984->5984/tcp      couchdb
6c5f40c6a78d        hyperledger/fabric-ca:x86_64-1.0.1       "sh -c 'fabric-ca-..."   About a minute ago   Up About a minute
 0.0.0.0:7054->7054/tcp                          ca.org1.example.com
65677ef15350        hyperledger/fabric-orderer:x86_64-1.0.1  "orderer"                About a minute ago   Up About a minute
 0.0.0.0:7050->7050/tcp                          orderer.example.com
```

Figure 3.5: Blockchain Network Containers Running Locally Before Chaincode Instantiation

3.4.2 Blockchain Network Modeling

The general architecture of the model consists of a blockchain runtime, which is composed of the infrastructure layer discussed in the previous section, and a network that executes the logic of our trust model. Hyperledger offers a blockchain development environment known as Hyperledger Composer. We used Composer to define the logic of our model. As there are different types of participants, they are assigned different levels of access to transactions based on their roles. For example, Alpha nodes, have the ability to access instances of network participants and update system parameters. IoT devices exchange information and have the ability to rate that transaction. Any involved parties that interact with the blockchain network must be registered as participants with the appropriate role assigned.

We defined the following in our model prototype:

- **Participants:** Participants in our network include IoT devices that are registered as members of the blockchain network and Alpha nodes. Alpha nodes can act as network maintainers (endorsing and committing peers) and store replicated copies of the blockchain ledger.

- **Transactions:** Transaction are represented by any interactions or information exchanges between the network's participants. This includes chaincode invocations, endorsements, trust evaluations, and data transmissions within IoT devices. Transactions are defined in JavaScript.
- **Events**: Events are emitted after a transaction is submitted to notify the network participants of the transaction's result. Events allow external applications to subscribe to them.

The Composer tool uses its own modeling language for the definition of entities. All the entities above, participants, transactions and events, are defined in the model's .cto file, included in Appendix A.1. The components involved in developing our model prototype through Composer are illustrated in Figure 3.6. Yeoman [5] can be used to generate an Angular[6] web application template using Composer files. The web application can be used to interact with the underlying fabric network. Angular web application is not part of our prototype; however, we discuss its usage later in our future work in Chapter 4 of this thesis.

Access control rules are defined as well. All files are then packaged into one archive for deployment in the blockchain runtime environment. Figure 3.7 shows the files defined for our prototype.

Model Participants

The prototype supports two type of participants: Alpha Nodes and IoT Devices. For a sample list of the network participants in our prototype, please refer to Figure A.1 in Appendix A. In this subsection, we discuss the role and attributes of each.

---

[5] yeoman is a framework for generation of dynamic web applications: http://yeoman.io/

[6] Angular is an open-source front-end web application platform: https://angular.io/

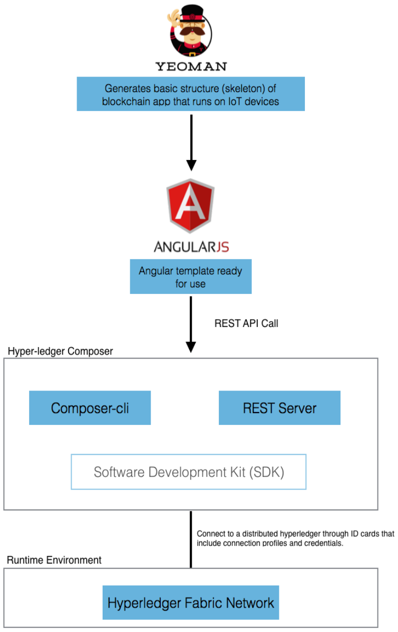

YEOMAN
Generates basic structure (skeleton) of blockchain app that runs on IoT devices
ANGULARJS
Angular template ready for use
REST API Call
Hyper-ledger Composer
Composer-cli
REST Server
Software Development Kit (SDK)
Connect to a distributed hyperledger through ID cards that include connection profiles and credentials.
Runtime Environment
Hyperledger Fabric Network

Figure 3.6: Fabric Composer Components [2]

Alpha Nodes:

Participants of type Alpha Node act as network administrators, as they are assumed to be capable and privileged nodes. Multiple Alpha Nodes can exist on the same network. Each Alpha Node participant bears the following attributes:

1. ID: A unique identification string to identify the Alpha Node on the network. The ID attribute takes the format of the word 'Alpha' + an integer number signifying the sequence. For example: *Alpha1*.
2. trustScore: Alpha nodes are participants with a trust score that always defaults to 100. This is due to their role that serves as administrators of the network. Their trust score cannot be changed.
3. sysThreshold: The threshold attribute is an integer variable holding a pre-defined system parameter that is used to adjust the network's strictness during trust evaluation.

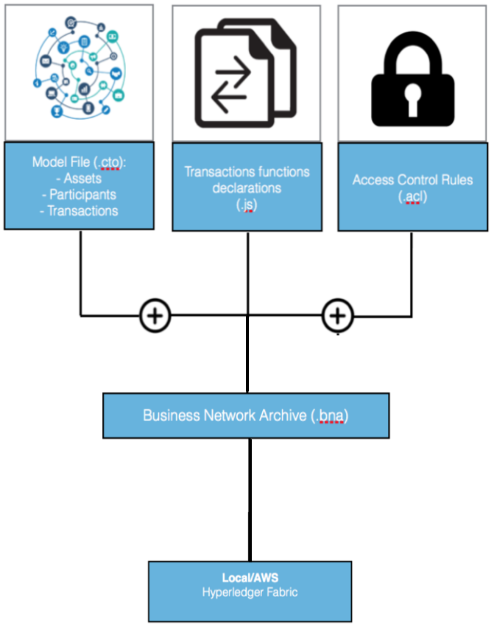

Model File (.cto):
- Assets
- Participants
- Transactions
Transactions functions declarations (.js)
Access Control Rules (.acl)
Business Network Archive (.bna)
Local/AWS
Hyperledger Fabric

Figure 3.7: Files Comprising The IoT Fabric Trust Model

The model allows for three levels of difficulty:

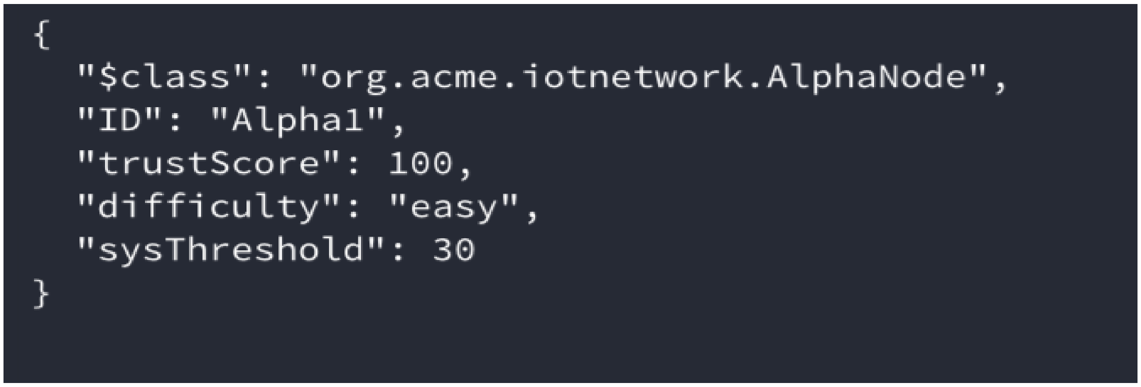
- Easy: When 'easy' difficulty level is set, the model evaluates devices' trust scores against a threshold set to 30. Only transactions submitted by devices with trust scores exceeding that threshold value are allowed.
- Medium: A more strict setting with a threshold set to 50.
- Strict: An extremely strict model evaluation setup allowing devices with a balance of trust points (trust score) of 70 or above to submit transactions on the network.

4. difficulty: Determines the difficulty level as described above. By default, the model's difficulty is set to *Medium*, with a threshold of value *50*.

Attributes of a sample participant of type Alpha Node is shown in Figure 3.8.

IoT Devices:

Each IoT Device participant bears the following attributes:



```
{
  "$class": "org.acme.iotnetwork.AlphaNode",
  "ID": "Alpha1",
  "trustScore": 100,
  "difficulty": "easy",
  "sysThreshold": 30
}
```

Figure 3.8: Network Participant of Role: Alpha Node

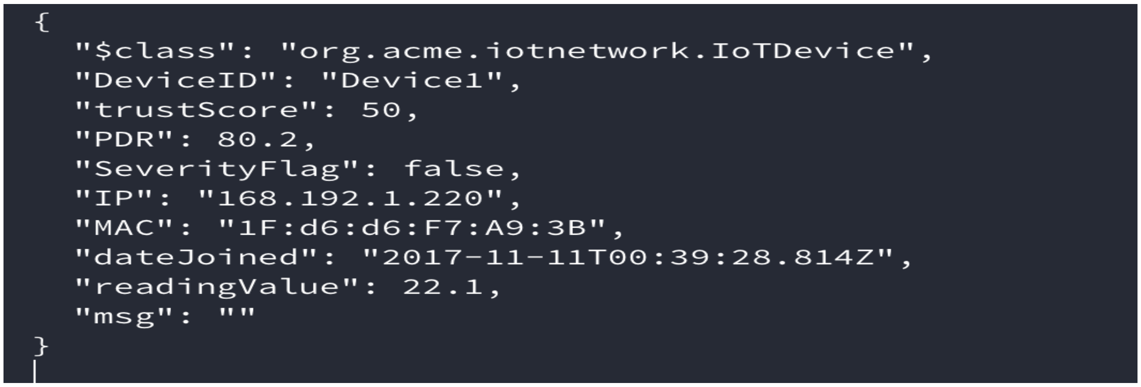
1. DeviceID: Unique identifier for each IoT device of String data type. Takes the format of the word 'Device' + an integer number signifying the sequence. For example: *Device1*.
2. PDR: The PDR represents the current Packet Delivery Rate observed by the network. We assume that devices with higher PDR values are most likely to be deemed healthy and trustworthy. The attribute holds a value of data type Double. For example: 77.3.
3. trustScore: The trust score attribute is an integer that holds available balance of trust points used for trust evaluation. The attribute is initialized to 50 at node enrollment and ranges between 0-100.
4. IP: The IP address of the IoT device.
5. Mac: The MAC address of the IoT device.
6. readingValue: Holds a simulated value of temperature reading of data type Double for other devices to inquire about. For example: 22.1.
7. SeverityFlag[7]: *Boolean* variable that indicates whether transmitted data is confidential and processed on the edge level or not. The attribute bears two values: true or false.

---

[7] This attribute is currently not used in the prototype. However, we discuss its significance in Chapter 4.

8. DateJoined: Records the date and time of device enrollment to the network, is of Date data type.
9. msg: Holds messages generated by recently executed transactions, is of String data type.

Attributes of a sample participant of type IoT Device is shown in Figure 3.9.



```
{
  "$class": "org.acme.iotnetwork.IoTDevice",
  "DeviceID": "Device1",
  "trustScore": 50,
  "PDR": 80.2,
  "SeverityFlag": false,
  "IP": "168.192.1.220",
  "MAC": "1F:d6:d6:F7:A9:3B",
  "dateJoined": "2017-11-11T00:39:28.814Z",
  "readingValue": 22.1,
  "msg": ""
}
```

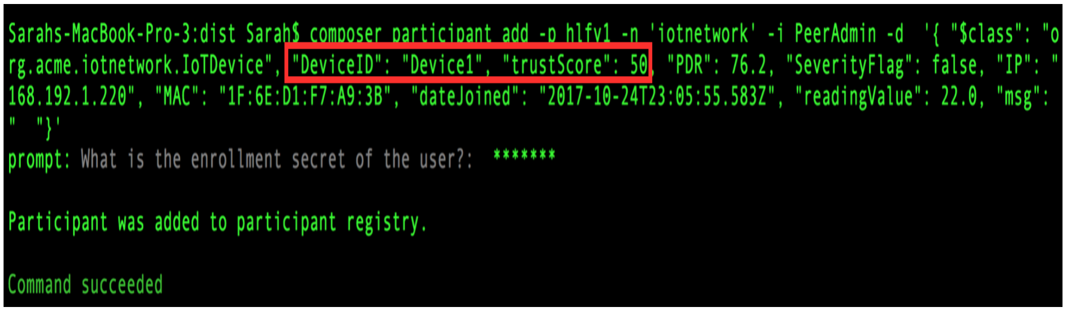
Figure 3.9: Network Participant of Role: IoT Device

Model Transactions and Events

Transaction behaviour occurring on the network is defined by functions that implement the logic of the model. The functions are written in JavaScript files and act as the blockchain network's smart contracts. After integration with the blockchain runtime, they are transformed to chaincode that can be instantiated on a peer. Our model supports the following functions:

1. Add Participant:

   The *Add Participant* function registers a new participant to the network. It is handled by the CA and requires sending a unique identifier value. An example for adding an IoT Device participant is shown in Figures 3.10. The participant is initialized with a default trust score of 50 points.



```
Sarahs-MacBook-Pro-3:dist Sarah$ composer participant add -p hlfv1 -n 'iotnetwork' -i PeerAdmin -d  '{ "$class": "o
rg.acme.iotnetwork.IoTDevice", "DeviceID": "Device1", "trustScore": 50, "PDR": 76.2, "SeverityFlag": false, "IP": "
168.192.1.220", "MAC": "1F:6E:D1:F7:A9:3B", "dateJoined": "2017-10-24T23:05:55.583Z", "readingValue": 22.0, "msg":
"  "}'
prompt: What is the enrollment secret of the user?:  *******

Participant was added to participant registry.

Command succeeded
```

Figure 3.10: Adding a New Participant: Device1

2. Issue Identity:

   The *Issue Identity* function uses the CA to issue a cryptographic identity for a given participant. A *userID* and *userSecret* are generated as well. The status of an identity can only be one of the following:

   (a) Issued: The *Issued* status indicates a newly created identity that has not been used yet by the participant.

   (b) Activated: The *Activated* status indicates that the identity has been used by the participant to submit transactions on the network.

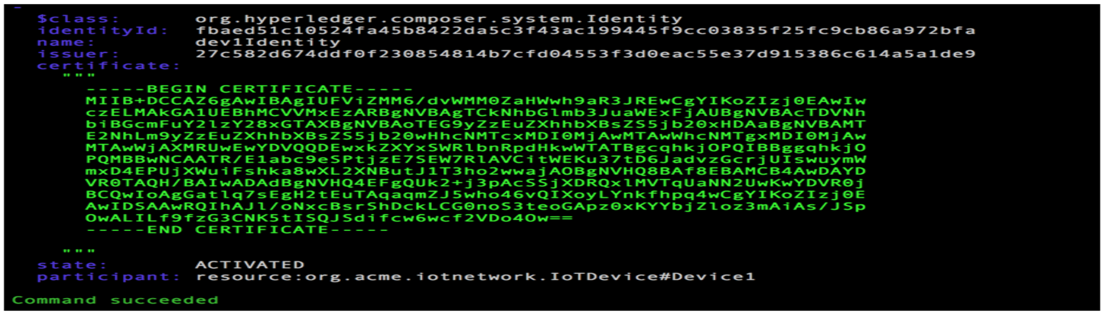
   (c) Revoked: The *Revoked* status indicates that the identity is no longer accepted as valid to participate on the network. A participant with a revoked identity can submit transactions; however, the network will reject it.

3. Revoke Identity:

   The *Revoke Identity* function revokes permissions and access rights of a given participant. This allows the blockchain network to reject all submitted transactions made by unauthorized participants. Figure 3.11 shows the status for an IoT Device participant, Device1, before and after identity revocation.

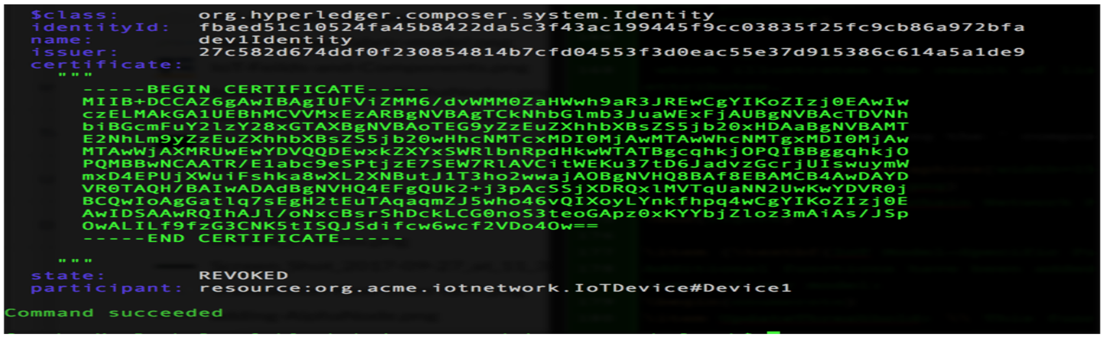
4. UpdateThreshhold:

   This function can be submitted by administrator users only. Alpha node participants



```
-
  $class:        org.hyperledger.composer.system.Identity
  identityId:    fbaed51c10524fa45b8422da5c3f43ac199445f9cc03835f25fc9cb86a972bfa
  name:          dev1Identity
  issuer:        27c582d674ddf0f230854814b7cfd04553f3d0eac55e37d915386c614a5a1de9
  certificate:
    """
      -----BEGIN CERTIFICATE-----
      MIIB+DCCAZ6gAwIBAgIUFViZMM6/dvWMM0ZaHWwh9aR3JREwCgYIKoZIzj0EAwIw
      czELMAkGA1UEBhMCVVMxEzARBgNVBAgTCkNhbGlmb3JuaWExFjAUBgNVBAcTDVNh
      biBGcmFuY2lzY28xGTAXBgNVBAoTEG9yZzEuZXhhbXBsZS5jb20xHDAaBgNVBAMT
      E2NhLm9yZzEuZXhhbXBsZS5jb20wHhcNMTcxMDI0MjAwMTAwWhcNMTgxMDI0MjAw
      MTAwWjAXMRUwEwYDVQQDEwxkZXYxSWRlbnRpdHkwWTATBgcqhkjOPQIBBggqhkjO
      PQMBBwNCAATR/E1abc9eSPtjzE7SEW7RlAVCitWEKu37tD6JadvzGcrjUIswuymW
      mxD4EPUjXWuiFshka8wXL2XNButJ1T3ho2wwajAOBgNVHQ8BAf8EBAMCB4AwDAYD
      VR0TAQH/BAIwADAdBgNVHQ4EFgQUk2+j3pAcSSjXDRQxlMVTqUaNN2UwKwYDVR0j
      BCQwIoAgGatlq7sEgH2tEuTAqaqmZJ5who46vQIXoyLYnkfhpq4wCgYIKoZIzj0E
      AwIDSAAwRQIhAJl/oNxcBsrShDckLCG0noS3teoGApz0xKYYbjZloz3mAiAs/JSp
      OwALILf9fzG3CNK5tISQJSdifcw6wcf2VDo4Ow==
      -----END CERTIFICATE-----

    """
  state:         ACTIVATED
  participant:   resource:org.acme.iotnetwork.IoTDevice#Device1

Command succeeded
```

A: Identity Activated After Submitting a Transaction on The Network



```
  $class:        org.hyperledger.composer.system.Identity
  identityId:    fbaed51c10524fa45b8422da5c3f43ac199445f9cc03835f25fc9cb86a972bfa
  name:          dev1Identity
  issuer:        27c582d674ddf0f230854814b7cfd04553f3d0eac55e37d915386c614a5a1de9
  certificate:
    """
      -----BEGIN CERTIFICATE-----
      MIIB+DCCAZ6gAwIBAgIUFViZMM6/dvWMM0ZaHWwh9aR3JREwCgYIKoZIzj0EAwIw
      czELMAkGA1UEBhMCVVMxEzARBgNVBAgTCkNhbGlmb3JuaWExFjAUBgNVBAcTDVNh
      biBGcmFuY2lzY28xGTAXBgNVBAoTEG9yZzEuZXhhbXBsZS5jb20xHDAaBgNVBAMT
      E2NhLm9yZzEuZXhhbXBsZS5jb20wHhcNMTcxMDI0MjAwMTAwWhcNMTgxMDI0MjAw
      MTAwWjAXMRUwEwYDVQQDEwxkZXYxSWRlbnRpdHkwWTATBgcqhkjOPQIBBggqhkjO
      PQMBBwNCAATR/E1abc9eSPtjzE7SEW7RlAVCitWEKu37tD6JadvzGcrjUIswuymW
      mxD4EPUjXWuiFshka8wXL2XNButJ1T3ho2wwajAOBgNVHQ8BAf8EBAMCB4AwDAYD
      VR0TAQH/BAIwADAdBgNVHQ4EFgQUk2+j3pAcSSjXDRQxlMVTqUaNN2UwKwYDVR0j
      BCQwIoAgGatlq7sEgH2tEuTAqaqmZJ5who46vQIXoyLYnkfhpq4wCgYIKoZIzj0E
      AwIDSAAwRQIhAJl/oNxcBsrShDckLCG0noS3teoGApz0xKYYbjZloz3mAiAs/JSp
      OwALILf9fzG3CNK5tISQJSdifcw6wcf2VDo4Ow==
      -----END CERTIFICATE-----

    """
  state:         REVOKED
  participant:   resource:org.acme.iotnetwork.IoTDevice#Device1
Command succeeded
```

B: Identity Status changes to ‘Revoked’ and Device1 Cannot Participate on The Network

Figure 3.11: Identity Revocation for Participant: Device1

can submit this transaction. It facilitates management of the networks’ strictness when evaluating trust. For our prototype, we start the network with threshold difficulty set to *easy*, as new devices are initialized with a trust balance of 50. We then trigger transactions between the devices to create interactions between them and in result, their balance of trust points (trust score) will be updated. After a period of time and as more devices join the network, the level of difficulty is set to *Medium*.

5. populateParticipantRecord:

   This function can be submitted by administrator users only. It is used to update attributes of participants.

6. updateDevicePDR:

   This function can be submitted by administrator users only. It is used to update PDR values of participants. For the purpose of the prototype, devices PDR values are assumed to be pulled automatically from an underlaying IoT network.

7. readTemperature:

   A function to simulate information exchange of a temperature reading between two devices. This function triggers the next function.

8. rewardOrPunish:

   This function is triggered to evaluate the device in question, which is providing a service or information. For example, if Device1 is requesting the current temperature from Device2, then Device2 is the one subject to evaluation. The transaction performs an evaluation using the available balance of trust points against a pre-defined threshold and updates the device's balance of trust points by either rewarding or punishing it. Afterwards, the *UpdateBalanceOfTurstPoints* event is triggered.

9. UpdateBalanceOfTurstPoints (event):

   An event that is emitted after the *rewardOrPunish* function is executed. It writes the updated trust score to a participant registry.

The first three functions are built in the Fabric Composer[8] tool. Code written to define the remaining functions is included in Appendix A.2.

Access Control Rules

An access control language is used to enable defining access control rules for both participants and transactions that take place on the blockchain network. Defining rules allows for determining which participants are permitted to create, read, update or delete elements within the blockchain network. To enforce access control over the network's resources, we define the following rules:

---

[8] https://hyperledger.github.io/composer/

1. PartOfNetwork Rule:

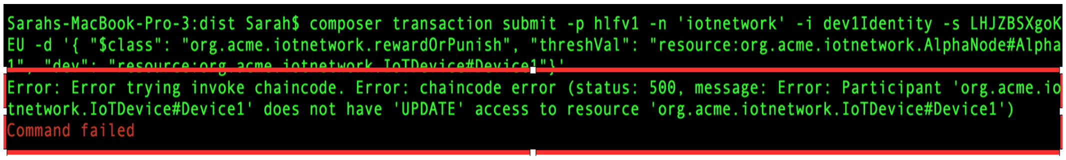
   Read only is granted over the network for IoT devices.

2. knowOfAll Rule:

   Devices can know of other devices connected to the network. They can ping the network as well.

3. NoSelfUpdate Rule:

   An IoT Device cannot update its own attributes, including its own trustscore. Figure 3.12 shows Device1, an IoT Device Participant, attempting to update its own trust score using the *rewardOrPunish* transaction. The transaction fails.



```
Sarahs-MacBook-Pro-3:dist Sarah$ composer transaction submit -p hlfv1 -n 'iotnetwork' -i dev1Identity -s LHJZBSXgoK
EU -d '{ "$class": "org.acme.iotnetwork.rewardOrPunish", "threshVal": "resource:org.acme.iotnetwork.AlphaNode#Alpha
1", "dev": "resource:org.acme.iotnetwork.IoTDevice#Device1"}'
Error: Error trying invoke chaincode. Error: chaincode error (status: 500, message: Error: Participant 'org.acme.io
tnetwork.IoTDevice#Device1' does not have 'UPDATE' access to resource 'org.acme.iotnetwork.IoTDevice#Device1')
Command failed
```

Figure 3.12: Device1 Failed Attempt in Updating its Own Trust Score

4. historianAccess Rule:

   Allows IoT devices to read and create historian records referencing their transactions. This allows for the creation of a new record after devices submit a transaction on the network.

5. DevSubmitTransaction Rule:

   Participants of type ‘IoTDevice’ can submit a transaction to update other devices’ trust scores.

6. DevUpdaterustScores Rule:

   Participants of type ‘IoTDevice’ can read and update trustScores of other IoT devices, only through the rewardOrPunish transaction.

7. DevReadTemperature Rule:

   Participants of type ‘IoTDevice’ can retrieve temperature readings from other IoT devices.

8. AlphaNodeAllAccess Rule:

   All access is granted over all network resources to ‘AlphaNode’.

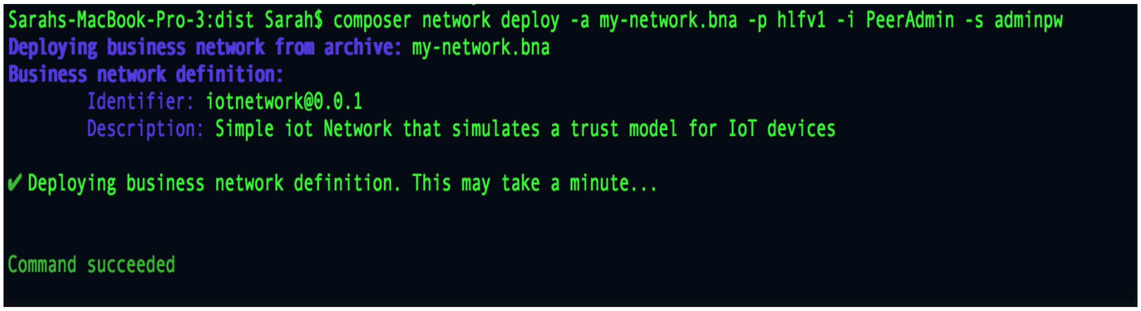
9. RevokeAccess Rule:

   Alpha Nodes are permitted to revoke permissions given to an IoT device if its trust score is equal to zero.

For complete definitions of the above access control rules, please refer to Appendix A.3.

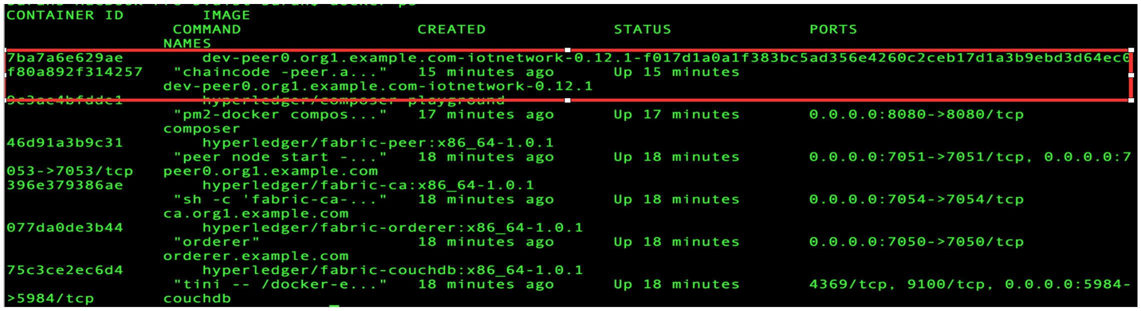
3.4.3 Integration

In the first implementation phase, we discussed the setup process of the hyperledger fabric bare environment running locally and had several network components running in docker containers. In the second phase, we defined the logic of the fabric network that is executed through chaincode using hyperledger fabric composer modeling language, and specified the approach of trust evaluation. Finally, we integrate the two by creating a hyperledger fabric connection profile. The connection profile helps point the modeled network to the underlying containers and their local ports. Contents of the connection file are available in Appendix A.4. Once the connection profile is set, we deploy the modeled network to the blockchain runtime. This enables instantiations of our logic (chaincode) on the peer node. All trust evaluations are performed by a peer node on the network. An example of a successful network deployment is demonstrated in Figure 3.13. Figure 3.14 shows peer containers with chaincodeID, indicating their capability of chaincode execution.



```
Sarahs-MacBook-Pro-3:dist Sarah$ composer network deploy -a my-network.bna -p hlfv1 -i PeerAdmin -s adminpw
Deploying business network from archive: my-network.bna
Business network definition:
        Identifier: iotnetwork@0.0.1
        Description: Simple iot Network that simulates a trust model for IoT devices

✔ Deploying business network definition. This may take a minute...


Command succeeded
```

Figure 3.13: Successful Network Model Deployment



```
CONTAINER ID        IMAGE
                 COMMAND                  CREATED             STATUS              PORTS
                NAMES
7ba7a6e629ae        dev-peer0.org1.example.com-iotnetwork-0.12.1-f017d1a0a1f383bc5ad356e4260c2ceb17d1a3b9ebd3d64ec0
f80a892f314257   "chaincode -peer.a..."   15 minutes ago      Up 15 minutes
                dev-peer0.org1.example.com-iotnetwork-0.12.1
9e3ac4bfdde1        hyperledger/composer-playground
                 "pm2-docker compos..."   17 minutes ago      Up 17 minutes       0.0.0.0:8080->8080/tcp
                composer
46d91a3b9c31        hyperledger/fabric-peer:x86_64-1.0.1
                 "peer node start -..."   18 minutes ago      Up 18 minutes       0.0.0.0:7051->7051/tcp, 0.0.0.0:7
053->7053/tcp   peer0.org1.example.com
396e379386ae        hyperledger/fabric-ca:x86_64-1.0.1
                 "sh -c 'fabric-ca-..."   18 minutes ago      Up 18 minutes       0.0.0.0:7054->7054/tcp
                ca.org1.example.com
077da0de3b44        hyperledger/fabric-orderer:x86_64-1.0.1
                 "orderer"                18 minutes ago      Up 18 minutes       0.0.0.0:7050->7050/tcp
                orderer.example.com
75c3ce2ec6d4        hyperledger/fabric-couchdb:x86_64-1.0.1
                 "tini -- /docker-e..."   18 minutes ago      Up 18 minutes       4369/tcp, 9100/tcp, 0.0.0.0:5984-
>5984/tcp       couchdb
```

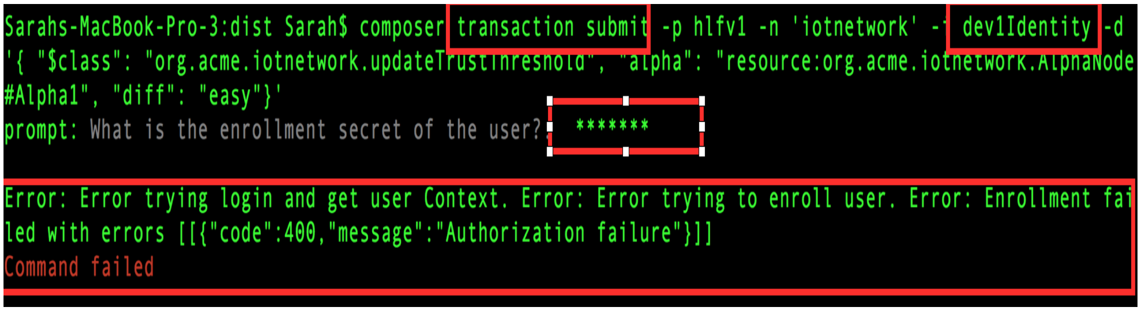
Figure 3.14: Fabric Network Containers Running Locally After Chaincode Instantiation

3.5 Implementation Results

In the previous sections, we introduced a prototype implementing a trust model for IoT using blockchains. To implement our model, we used the open source Hyperledger Fabric tool. The resulting model limits enrollment of new devices and identity management to CAs. Unlike a typical IoT network, all devices without any exceptions must be registered by the CA in order to be part of the network. Our model helps protect against multiple external and internal attacks including Sybil, message replay, wormwhole, and bad and good mouthing attacks discussed in Chapter 2. The model is resilient for several reasons:

- Devices permissioned membership:

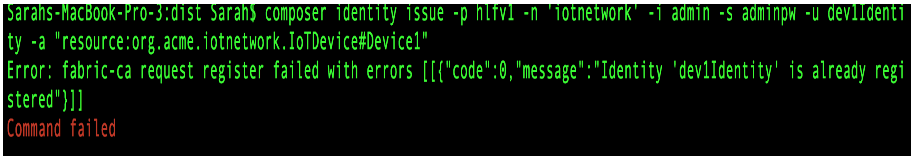
  Various information about the network participants, including their IDs, balance of trust points and IP addresses are recorded by our model. Participants are required to possess blockchain identities in order to participate and submit transactions on the network. This provides a proof of identity, similar to holding a passport, for example [32]. Figure 3.15 shows an attempt to submit a transaction by an added participant, before receiving its identity; the transaction fails.



```
Sarahs-MacBook-Pro-3:dist Sarah$ composer transaction submit -p hlfv1 -n 'iotnetwork' -i dev1Identity -d
'{ "$class": "org.acme.iotnetwork.updateTrustThreshold", "alpha": "resource:org.acme.iotnetwork.AlphaNode
#Alpha1", "diff": "easy"}'
prompt: What is the enrollment secret of the user? *******

Error: Error trying login and get user Context. Error: Error trying to enroll user. Error: Enrollment fai
led with errors [[{"code":400,"message":"Authorization failure"}]]
Command failed
```

Figure 3.15: Submitting a Transaction Without an Identity

Enrolled participants can get approval of submitted transactions only if those participants are bound to a valid identity. Attempting to issue an identity for non-existing participants results in failure. Moreover, attempting to issue a duplicated identity will also result in failure, as shown in Figure 3.16.



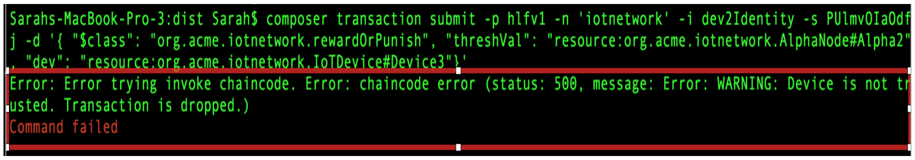
```
Sarahs-MacBook-Pro-3:dist Sarah$ composer identity issue -p hlfv1 -n 'iotnetwork' -i admin -s adminpw -u dev1Identi
ty -a "resource:org.acme.iotnetwork.IoTDevice#Device1"
Error: fabric-ca request register failed with errors [[{"code":0,"message":"Identity 'dev1Identity' is already regi
stered"}]]
Command failed
```

Figure 3.16: Attempting Identity Duplication

Through this approach of identity management, rogue devices cannot issue their own identities nor impersonate identities of other devices on the network. Issuing *enrollment certificates* for participants as their identity document allows participants to cryptographically sign submitted transactions on the blockchain network [32]. Identities and their corresponding participant mappings are stored in an Identity Registry. Once the CA issues a new identity for a participant, the mapping is added to the registry. The identity of the participant is *Activated* once a transaction is submitted with that identity. Upon submitting a transaction by a device, the blockchain searches the identity registry for a valid mapping for the used identity. This is achieved by looking up the hash of the enrollment certificate. The corresponding participant is retrieved once a mapping for the identity is found. After that, the retrieved participant becomes the *Current Participant*, which the blockchain evaluates the defined access control rules against [32]. Through this approach, an adversary cannot force bad ratings for certain devices to deny them service or to ruin their reputation on the IoT network. Additionally, forcing IoT devices to interact with valid identities only allows for the ability to verify the source of submitted transactions.

- Trust evaluation:

  Devices can check other devices' trust score at any given time. Devices must have sufficient balance of trust points in order to participate in the network activities. As a result, we can detect internal rogue devices. For example, Device2, holding a trust score of 50, proposes a transaction with Device3. The current trust score of Device3 is *30* and the difficulty of the model is set to *Medium*. This setup allows devices with trust scores of 50 or higher to be trusted. The above transaction is rejected because Device3 does not possess sufficient trust points to be recognized as trusted. The result of that transaction proposal is shown in Figure 3.17. Unlike the reviewed TRM model [7], trust of IoT devices in our proposed model is constantly evaluated with each transaction. This is to address the possibility that a device can turn malicious after a period of trusted behaviour.



```
Sarahs-MacBook-Pro-3:dist Sarah$ composer transaction submit -p hlfv1 -n 'iotnetwork' -i dev2Identity -s PUlmvOIaOdf
j -d '{ "$class": "org.acme.iotnetwork.rewardOrPunish", "threshVal": "resource:org.acme.iotnetwork.AlphaNode#Alpha2"
, "dev": "resource:org.acme.iotnetwork.IoTDevice#Device3"}'
Error: Error trying invoke chaincode. Error: chaincode error (status: 500, message: Error: WARNING: Device is not tr
usted. Transaction is dropped.)
Command failed
```

Figure 3.17: Trust Evaluation

- Input Validation:

Input of transactions is validated by endorsing peers ahead of committing. Validating transaction certificate prevents message replay attacks, as endorsing peers will detect if the same transaction has been submitted previously. This also helps in detection of good and bad mouthing attacks.

- Immune History and Secure Storage:

  Our model prevents modifications of previous transactions, including devices' reward and punishment. Attempts to modify previously submitted transactions to manipulate trust scores will fail due to the append-only feature of the ledger. A single change of a transaction hash that is stored in the blockchain results in a different root MHT hash. This contradiction immediately invalidates the block. In addition, IoT devices on the networks do not participate in keeping copies of the ledger. The ledger is managed by peer nodes on the blockchain.

- Single Channel:

  The model is configured to be contained in a single network channel. It is possible to run the blockchain on different channels; however, our model limits all transactions to one channel. As a result, we can decrease chances of wormhole attacks.

- Light-weight Transaction:

  Our model takes into account that a huge portion of IoT devices are limited in resources. Therefore, the model limits transactions made by IoT devices while interacting with the blockchain to simple RESTfull requests. The protocol is supported by the majority of IoT devices, which allows our model to address the heterogeneity of IoT as well.

3.6 Conclusions

In this chapter, we introduced our proposed IoT trust model that allows transaction exchange only among IoT devices that are determined to be trustworthy. The model protects the identities of its IoT devices and helps protect the integrity of transactions. We discussed our approach for building a prototype that implements a proof-of-concept of portions of our proposed model. In our model, all instances that are involved in the IoT network interactions can be identified and registered. Every enrolled device keeps cryptographic key pairs generated by a trusted CA. Transactions are proposed, simulated and assessed for trustworthiness before the transaction is committed. Hence, it is unlikely for identities to be impersonated or duplicated. All transactions and interactions are written to a public ledger, where transactions are only allowed to be appended to the ledger. As a result, modifying historical interactions to effect trust scoring of certain devices on the network is not allowed.

Chapter 4

Conclusions and Future Work

The nature of IoT requires interconnection with different types of devices. Presumption of trust exists in many IoT networks. As a result, IoT networks become vulnerable to various attacks that can disrupt the network and allow for manipulation of devices' reputations. As more *things* connect to the IoT network, it becomes more difficult to scale security. Devices on the IoT network can be tricked into revealing information or providing trusting feedback. In return, this results is violating the integrity of transmitted data, as well as its source. This consequently compromises the whole effectiveness of the IoT network.

In this thesis, we introduced an IoT trust model that utilizes permissioned blockchains to dynamically evaluate trust of IoT devices. The permissioned architecture facilitates secure identity and key management to prevent various network attacks. Moreover, the proposed model dynamically updates trust scores for participants following interactions between two participants. This allows for a level of trust, where only trusted devices are allowed to participate by submitting transactions on the network. We also introduced a proof-of-concept prototype that implements our proposed IoT trust model. In this chapter, we conclude this thesis and provide recommendations for future work.

4.1 Conclusions

Trust in most IoT networks is presumed implicitly. Trust issues in IoT environments are one of the main causes of different attacks including Sybil, reply, bad mouthing and good mouthing attacks. Sybils in the IoT network hold replicated identities. By controlling multiple identities that appear valid, adversaries can disrupt the network and manipulate reputations of trusted devices. An important aspect of securing IoT networks is maintaining a trusted IoT environment. A trusted IoT environment ensures that only authenticated and authorized devices are able to participate in the IoT network's activities. This can be accomplished through implementation of PKI schemes to guarantee confidentiality, authenticity of identities and access control rules. The wireless nature of IoT makes it possible for adversaries to capture network packets generated by a trusted IoT device to imitate its behaviour and appear trustworthy. Therefore, verifying identities and making sure that transactions are digitally signed by the correct device is important. Additionally, a trusted IoT environment does not rely on initial authentication as a permanent indication of trust. This is because devices in an IoT network can turn rogue upon compromise by adversaries.

In Chapter 1, we have mentioned multiple breaches that have proven IoT devices to be vulnerable to many types of attacks. Common security defenses in place have proven insufficient against preventing basic and unsophisticated attacks on smart devices. Therefore, constant refinement of trust is crucial to detect malicious nodes. A secure and trusted IoT network does not only possess the ability to manage identities and constantly evaluate trust, but it should also record and protect the integrity of all transactions on the IoT network. This includes transactions that result in the update of trust scores of IoT devices. As these trust scores can be manipulated by an adversary to influence reputations of IoT devices. In addition to maintaining a secure storage and a tamper-proof transaction history, a model is needed to fit the varying types, resources, and operating systems of different devices in order to adjust to IoT's heterogeneity. In this work, we were able to show that such model is possible.

All the ideal IoT trusted environment properties mentioned above are present in our proposed IoT trust model and implemented in our prototype. Our model uses permissioned blockchains' architecture to securely enroll IoT devices and issue their identities. Every IoT device holds a blockchain identity that is verified a head of transaction execution. In addition to verifying that a submitted transaction carries a valid IoT device signature, it is evaluated for proper format. The submission timestamp is also checked to ensure the transaction has not been submitted before. This protects our model from being affected by message reply attacks. Through this validation, Sybil nodes cannot influent the network, as they cannot replicate this type of identity. If the transaction passed these checks, the blockchain uses defined logic to determine whether to approve or reject transaction; this is determined in trust evaluation.

Trust evaluation is the final transaction check that is performed by the validating peers. Unlike any of the existing IoT trust models, in our model, trust evaluation logic is implemented through smart contracts. Smart contracts are defined instantiated once at the model and cannot be modified. This means an adversary cannot change trust evaluation rules that have been already defined in the model. Distributed processing is performed by multiple blockchain *peers* that execute smart contracts. Smart contracts manage and determine the logic behind the blockchain's transactions. Invocation of defined functions results in a transaction on the blockchain network. Trust calculation and evaluation is performed through invocation of smart contracts functions that perform PNN classification for devices with a severity flag set to 1 and a quick trust evaluation for those with the flag set to 0. Devices' behaviour history and recommendations given by other devices on the IoT network are taken into consideration in trust calculations. Devices are rated and recommended based on their *trust scores*. Trust scores are updated by an IoT device directly after interaction or data exchange activities between two IoT devices. The requesting IoT device rewards the serving IoT device with high trust points, or punishes it with low trust points ranging between 0-5. Devices are deemed trustworthy if approved by multiple peers. If the transaction is not endorsed (device determined trustworthy) by enough peers, it is rejected and dropped. The model constantly refines trust scores after each transaction to detect any malicious misbehaving devices. Additionally, the model tackles the cold start problem through handling trust initialization for newly added devices by predicting their trust score based on devices' collected information.

To protect the integrity of submitted rewards and punishments, all transactions are securely stored in the blockchain's ledger, which does not allow modifications. In order for an adversary to modify a single transaction, they have to recalculate all hashes of all previous blocks tracing back to the first block. Additionally, copies of the ledger are managed and stored on peer nodes only.

Our model takes into consideration that a majority of IoT devices can be limited in resources, such as bandwidth, battery and processing power. Therefore, the model limits transactions made by IoT devices while interacting with the blockchain to simple RESTfull http requests. The protocol is supported by the majority of IoT devices, which allows our model to address the heterogeneity of IoT as well.

Through implementing a proof-of-concept of our IoT trust model prototype, we have shown that our model is capable of protecting IoT networks from various attacks. Using blockchains to evaluate trust in IoT networks can ensure integrity of transactions and authentication of identities of IoT devices on the network. Our proposed model is a suitable fit to the changing needs of IoT as it takes advantage of decentralized processing and provides collaboration across different types of nodes. This allows our model to to scale to fit any kind of IoT network, without being constrained to a certain context.

4.2 Future Work

In our prototype, all participating nodes run locally on one physical machine. Yet, by running different nodes in separate Docker containers, we are able to simulate a distributed blockchain network. Currently, one CA is handling MSP responsibilities. Those responsibilities include issuing enrollment certificates and facilitating registrations of new participants (IoT devices). This management of cryptographic keys can consume the network's resources and can possibly limit its scalability [34], as the fabric CA is the only authority capable of performing this task and the list of identities is updated frequently. However, in the future we can improve our model by adding subordinate CAs that inherit that authority. In other words, distributing key management. This can significantly decrease network overhead and improve performance and scalability. Our next step is to implement multiple CAs on the same sever. Additionally, we will add more peer nodes to the blockchain network for transaction processing. We will also use Yeoman applications to design a mobile application allowing IoT devices to interact directly with the blockchain through RESTful API requests.

In our prototype, we built a proof-of-concept for some of our model's key components. In the future, we will implement a distributed PNN for trustworthy devices' classification in blockchain smart contract. We will also implement the *severityFlag* we mentioned in Chapter 3 to provide different levels of security and minimize trust calculations for data that is publicly available. This can help obtain faster response times between devices and improved security for confidential data.

Appendix A

A.1 Model's CTO File

```
/**
 * My iot network
 */
namespace org.acme.iotnetwork

/*
 * An enumeration that defines the three levels of difficulty allowed for the trust model.
  ---> Easy setting allows for a blind-trust approach. Even devices that are low on trust points can be evaluated as trusted.
  ---> Medium setting allows for a less strict trust evaluation.
  ---> Strict setting allows for a very strict trust evaluation. Only devices wih high balance of trust points can submit transactions.
 */
enum difficultyLevel{
 o easy
 o medium
 o strict
}

/* asset threshold identified by thresholdseq{
   o String thresholdseq default='1'
   o String difficulty default='easy'
   o Integer sysThreshold default = 30
}
*/

participant IoTDevice identified by DeviceID {
     o String DeviceID
  o Integer trustScore default=50 range=[0,100]
```

```
    o Double PDR
    o Boolean SeverityFlag
    o String IP
    o String MAC
    o DateTime dateJoined
    o Double readingValue
    o String msg
}

/*
Alpah Nodes are trusted, capable nodes on the Fabric network with a maximum number of trust points.
These node manage the difficulty level of the model.
Multiple Alpha nodes can exist.
*/
participant AlphaNode identified by ID {
    o String ID
    o Integer trustScore default=100
    o String difficulty default='medium'
    o Integer sysThreshold default=50
}

/*
Admin function to update system threshold
*/
transaction updateTrustThreshold{
    --> AlphaNode alpha
    o difficultyLevel diff
}

/*
Admin function to update/populate participant (IoT device) info.
*/
transaction populateParticipantRecord{
    --> IoTDevice newDevice
    o Double newDevPDR
    o Boolean newDevSeverityFlag
    o String newDevIP
    o DateTime newDevdateJoined
    o Double newDevreadingValue
    o String newDevMac
}

/*
Update PDR value of an IoT device for testing.
*/
transaction updateDevicePDR{
    --> IoTDevice dev
    o Double devPDR
}

/*
```

```
Reward an iot device with + points or punish it - points after a transaction between two device.
*/
transaction rewardOrPunish{
    --> AlphaNode threshVal
    --> IoTDevice dev
}

/*
Simulate data exchange of temperature between two IoT devices*/
transaction readTemperature{
    --> IoTDevice dev
}

/*
Event triggered causing upfate of trust balance after reward of punishment.
*/
event UpdateBalanceOfTurstPoints {
    --> IoTDevice dev
}
```

A.2 Model's Logic

```
/*
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

/**
* Update Sys threshold
Admin function to update system threshold
* @param {org.acme.iotnetwork.updateTrustThreshold} th - the threshold (held by an alpha node) to be updated
* @transaction
*/


function updateTrustThreshold(th) {
    var oldThreshold = th.alpha.sysThreshold;
    //var newThreshold = th.newThresholdVal;
    var oldDiff = th.alpha.difficulty;
    var NewDiff = th.diff;

    console.log('Changing system threshold...');

    if ((oldDiff == 'easy' && NewDiff == 'medium') || (oldDiff == 'easy' && NewDiff == 'strict') || (oldDiff == 'medium' && NewDiff == 'strict')) {
        console.log('Increasing difficulty. Model is more strict');

    }
    else {
        console.log('Decreasing difficulty. Model is less strict');
    }


    if (NewDiff == 'easy')
        var newThreshold = 30;
    else if (NewDiff == 'medium')
        var newThreshold = 50;
    else
        var newThreshold = 70;


    // th.newThresholdVal =   newThreshold;

    console.log('Old threshold:' + oldThreshold);
    console.log('New threshold:' + newThreshold);

    th.alpha.sysThreshold = newThreshold;
    th.alpha.difficulty = NewDiff;
    return getParticipantRegistry('org.acme.iotnetwork.AlphaNode').then(
        function (participantRegistry) {
            return participantRegistry.update(th.alpha);
        });
}


/**
 * populate or update an IoT device participant record for debugging
 * @param {org.acme.iotnetwork.populateParticipantRecord} d - the device to be populated
 * @transaction
 */
function populateParticipantRecord(d) {
    var device = d.newDevice.deviceID;
    var pdr = d.newDevPDR;
    var flag = d.newDevSeverityFlag;
    var ip = d.newDevIP;
    var date = d.newDevdateJoined;
    var val = d.newDevreadingValue;
    var mac = d.newDevMac;


    d.newDevice.msg = " Populating participant: " + device + "Device
    information updated " + "Device PDR:" + pdr + "Device severity flag:" + flag + "Device IP address" + ip + "Joined on: " + date + "Holds value: " + val;
    d.newDevice.PDR = pdr;
    d.newDevice.SeverityFlag = flag;
    d.newDevice.IP = ip;
    d.newDevice.MAC = mac;
    d.newDevice.dateJoined = date;
    d.newDevice.readingValue = val;


    return getParticipantRegistry('org.acme.iotnetwork.IoTDevice').then(
        function (ParticipantRegistry) {
            return ParticipantRegistry.update(d.newDevice);
        });

}

/**
 * Update PDR value for an IoT device ( for testing)
 * @param {org.acme.iotnetwork.updateDevicePDR} d - the device to be updated
 * @transaction
 */
function updateDevicePDR(d) {
    var newpdr;
    newpdr = d.devPDR;

    d.dev.PDR = newpdr;

    return getParticipantRegistry('org.acme.iotnetwork.IoTDevice').then(
        function (ParticipantRegistry) {
            return ParticipantRegistry.update(d.dev);
        });
}


/**
 * read temperature value
 * @param {org.acme.iotnetwork.readTemperature} temp - the value of temp reading.
 * @transaction
 */
function readTemperature(temp) {
    var tempReading = temp.dev.readingValue;

    temp.dev.msg = "Temperature value captured by device:" + tempReading + ".";

    console.log('Temperature value captured by device:' + tempReading + ".");
}

/**
```

```
 * evaluate the trust of an iot device and reward or punish an IoT device with trust points
 * @param {org.acme.iotnetwork.rewardOrPunish} t - the trust score to be evaluated
 * @transaction
 */
function rewardOrPunish(t) {
    // Save the old value of the asset.
    var threshold = t.threshVal.sysThreshold;
    var oldValue = t.dev.trustScore;
    var pdrValue = t.dev.PDR;
    var updatePointsBy = 1;
    var trusted = false;


    if (pdrValue >= 70) {
        updatePointsBy = 10;
    }
    else if (pdrValue < 50 && pdrValue >= 30) {
        updatePointsBy = 5;
    }
    else {
        updatePointsBy = 1;
    }

    if (oldValue >= threshold) {

        // Reward the node for exceeding the trust threshold :
        // [1] Allow communication between Device1 & Device2
        // [2] check the device's PDR to determine how many points to give for reward
        // [3] and update the balance
        t.newValue = t.dev.trustScore + updatePointsBy;
        t.dev.trustScore = t.newValue;
        trusted = true;

        t.dev.msg = "Device trust score exceeds threshold. Trusting Device... Transaction proceeds. ";
        console.log('Device trust score exceeds threshold. Trusting Device... Transaction proceeds. ');
    }
    else {
        t.newValue = t.dev.trustScore - updatePointsBy;
        t.dev.trustScore = t.newValue;
        // Punish the node for having the trust threshold :
        // [1] Allow communication between Device1 & Device2
        // [2] check the device's PDR to determine how many points to give for reward
        // [3] and update the balance
        t.dev.msg = "Device trust score is below threshold.";
        throw new Error("WARNING: Device is not trusted. Transaction is dropped.");
    }

    return getParticipantRegistry('org.acme.iotnetwork.IoTDevice').then(
        function (ParticipantRegistry) {

            // emit a notification that trust balance has been updated
            var UpdateBalanceOfTurstPoints = getFactory().newEvent('org.acme.iotnetwork', 'UpdateBalanceOfTurstPoints');
            UpdateBalanceOfTurstPoints.dev = t.dev;
            emit(UpdateBalanceOfTurstPoints);

           return ParticipantRegistry.update(t.dev);
        });
}
```

A.3 Access Control Rules

```
/**
 * Access Control Rules *
 ==============================================================================
 * Rules governing access for participants of typr: IoT Devices
 ==============================================================================
 */

rule PartOfNetwork{
     description: "READ only is granted over network participants for IoT devices."
     participant: "org.hyperledger.composer.system.Participant"
     operation: READ
     resource: "org.hyperledger.composer.system.**"
     action: ALLOW
 }

rule knowOfAll {
     description: "Devices can know of other devices connected to the network."
     participant: "org.acme.iotnetwork.IoTDevice"
     operation: READ
     resource: "org.hyperledger.composer.system.Participant"
     action: ALLOW
 }

rule NoSelfUpdate{
    description: "An IoT Device cannot update its own attributes, including its own trustscore."
    participant(m): "org.acme.iotnetwork.IoTDevice"
    operation: CREATE, UPDATE, DELETE
    resource(v): "org.acme.iotnetwork.IoTDevice"
    condition: (v.getIdentifier() == m.getIdentifier())
    action: DENY
}
```

```

rule historianAccess{
    description: "Allow IoT devices to read and CREATE historian records referencing their transactions."
    participant: "org.acme.iotnetwork.IoTDevice"
    operation: READ, CREATE
    resource: "org.hyperledger.composer.system.HistorianRecord"
    action: ALLOW
}

rule DevSubmitTransaction{
     description: "Participants of type 'IoTDevice' can submit a transaction to update other devices' trust scores"
     participant: "org.acme.iotnetwork.IoTDevice"
     operation: READ, UPDATE, CREATE
     resource: "org.acme.iotnetwork.rewardOrPunish"
     action: ALLOW
 }

rule DevUpdaterustScores {
     description: "Participants of type 'IoTDevice' can  READ and update trustScores of other IoTdevices"
     participant: "org.acme.iotnetwork.IoTDevice"
     operation: READ, UPDATE
     resource: "org.acme.iotnetwork.IoTDevice"
     action: ALLOW
 }

rule DevReadTemperature{
     description: "Participants of type 'IoTDevice' can READ temperature readings from other IoTdevices"
     participant: "org.acme.iotnetwork.IoTDevice"
     operation: READ, CREATE
     resource: "org.acme.iotnetwork.readTemperature"
     transaction: "org.acme.iotnetwork.readTemperature"
     action: ALLOW
 }
//==========================================================================

 /**
 =========================================================================
 * Rules governing network access for participants of type: Alpha Node
 =========================================================================
 */

rule AlphaNodeAllAccess {
     description: "AllAccess is granted over everything to AlphaNodes."
     participant: "org.acme.iotnetwork.AlphaNode"
     operation: ALL
     resource: "org.hyperledger.composer.system.**"
     action: ALLOW
 }
```

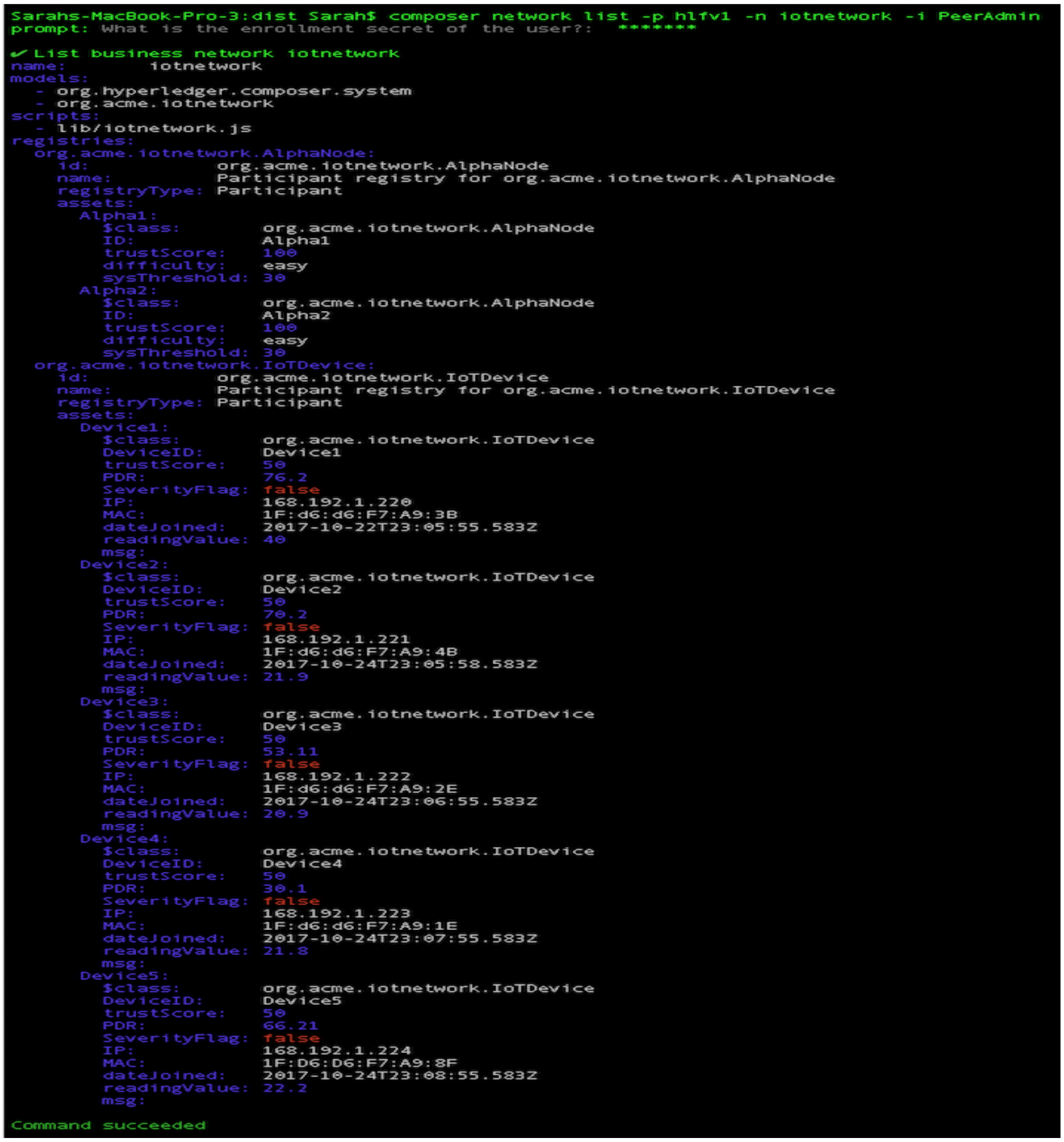
```


rule RevokeAccess{
    description: "Revoke privilges given to an IoT device "
    participant: "org.acme.iotnetwork.AlphaNode#Alpha1"
    operation: DELETE
    resource(d): "org.acme.iotnetwork.IoTDevice"
    condition: (d.trustScore==0)
    action: ALLOW
}
//============================================================================
```

A.4 Connection Profile (.json)

```
{
    "type": "hlfv1",
    "orderers": [
       { "url" : "grpc://localhost:7050" }
    ],
    "ca": { "url": "http://localhost:7054",
            "name": "ca.org1.example.com"
    },
    "peers": [
        {
            "requestURL": "grpc://localhost:7051",
            "eventURL": "grpc://localhost:7053"
        }
    ],
    "keyValStore": "/Users/Sarah/.composer-credentials",
    "channel": "composerchannel",
    "mspID": "Org1MSP",
    "timeout": "300"
}
```

A.5 IoT Blockchain Network Participants

The Figure below lists a sample of the network participants in our prototype.



```
Sarahs-MacBook-Pro-3:dist Sarah$ composer network list -p hlfv1 -n iotnetwork -i PeerAdmin
prompt: What is the enrollment secret of the user?:  *******

✔ List business network iotnetwork
name:       iotnetwork
models:
  - org.hyperledger.composer.system
  - org.acme.iotnetwork
scripts:
  - lib/iotnetwork.js
registries:
  org.acme.iotnetwork.AlphaNode:
    id:           org.acme.iotnetwork.AlphaNode
    name:         Participant registry for org.acme.iotnetwork.AlphaNode
    registryType: Participant
    assets:
      Alpha1:
        $class:       org.acme.iotnetwork.AlphaNode
        ID:           Alpha1
        trustScore:   100
        difficulty:   easy
        sysThreshold: 30
      Alpha2:
        $class:       org.acme.iotnetwork.AlphaNode
        ID:           Alpha2
        trustScore:   100
        difficulty:   easy
        sysThreshold: 30
  org.acme.iotnetwork.IoTDevice:
    id:           org.acme.iotnetwork.IoTDevice
    name:         Participant registry for org.acme.iotnetwork.IoTDevice
    registryType: Participant
    assets:
      Device1:
        $class:       org.acme.iotnetwork.IoTDevice
        DeviceID:     Device1
        trustScore:   50
        PDR:          76.2
        SeverityFlag: false
        IP:           168.192.1.220
        MAC:          1F:d6:d6:F7:A9:3B
        dateJoined:   2017-10-22T23:05:55.583Z
        readingValue: 40
        msg:
      Device2:
        $class:       org.acme.iotnetwork.IoTDevice
        DeviceID:     Device2
        trustScore:   50
        PDR:          70.2
        SeverityFlag: false
        IP:           168.192.1.221
        MAC:          1F:d6:d6:F7:A9:4B
        dateJoined:   2017-10-24T23:05:58.583Z
        readingValue: 21.9
        msg:
      Device3:
        $class:       org.acme.iotnetwork.IoTDevice
        DeviceID:     Device3
        trustScore:   50
        PDR:          53.11
        SeverityFlag: false
        IP:           168.192.1.222
        MAC:          1F:d6:d6:F7:A9:2E
        dateJoined:   2017-10-24T23:06:55.583Z
        readingValue: 20.9
        msg:
      Device4:
        $class:       org.acme.iotnetwork.IoTDevice
        DeviceID:     Device4
        trustScore:   50
        PDR:          30.1
        SeverityFlag: false
        IP:           168.192.1.223
        MAC:          1F:d6:d6:F7:A9:1E
        dateJoined:   2017-10-24T23:07:55.583Z
        readingValue: 21.8
        msg:
      Device5:
        $class:       org.acme.iotnetwork.IoTDevice
        DeviceID:     Device5
        trustScore:   50
        PDR:          66.21
        SeverityFlag: false
        IP:           168.192.1.224
        MAC:          1F:D6:D6:F7:A9:8F
        dateJoined:   2017-10-24T23:08:55.583Z
        readingValue: 22.2
        msg:

Command succeeded
```

Figure A.1: Sample Network Participants

References

[1] Y. B. Saied, A. Olivereau, D. Zeghlache, and M. Laurent, “Trust management system design for the internet of things: A context-aware and multi-service approach,” *Computers & Security*, vol. 39, pp. 351–365, 2013.

[2] Typical hyperledger composer solution architecture. Accessed on October 15, 2017. [Online]. Available: https://hyperledger.github.io/composer/introduction/solution-architecture.html

[3] G. Xu, Y. Ding, J. Zhao, L. Hu, and X. Fu, “Research on the internet of things (IoT),” *Sensors and Transducers*, vol. 160, no. 12, pp. 463–471, 12 2013.

[4] Samsung smart fridge leaves gmail logins open to attack. Accessed on November 25, 2017. [Online]. Available: https://www.theregister.co.uk/2015/08/24/smart_fridge_security_fubar/

[5] Intruders hack industrial heating system using backdoor posted online. Accessed on November 25, 2017. [Online]. Available: https://arstechnica.com/information-technology/2012/12/intruders-hack-industrial-control-system-using-backdoor-exploit/

[6] P. N. Mahalle, B. Anggorojati, N. R. Prasad, and R. Prasad, “Identity authentication and capability based access control (iacac) for the Internet of things,” *Journal of Cyber Security and Mobility*, vol. 1, no. 4, pp. 309–348, 2013.

[7] D. Chen, G. Chang, D. Sun, J. Li, J. Jia, and X. Wang, “TRM-IoT: a trust management model based on fuzzy reputation for Internet of things,” *Computer Science and Information Systems*, vol. 8, no. 4, pp. 1207–1228, 2011.

[8] S. Asiri and A. Miri, “An IoT trust and reputation model based on recommender systems,” in *Privacy, Security and Trust (PST), 2016 14th Annual Conference on Privacy, Security and Trust*. IEEE, 2016, pp. 561–568.

[9] C. Miller and C. Valasek, “Remote exploitation of an unaltered passenger vehicle,” Tech. Rep., 2015.

[10] D. Bandyopadhyay and J. Sen, “Internet of things: Applications and challenges in technology and standardization,” *Wireless Personal Communications*, vol. 58, no. 1, pp. 49–69, 2011.

[11] R. Roman, P. Najera, and J. Lopez, “Securing the Internet of things,” *Computer*, vol. 44, no. 9, pp. 51–58, Sept 2011.

[12] Y. Yu, K. Li, W. Zhou, and P. Li, “Trust mechanisms in wireless sensor networks: Attack analysis and countermeasures,” *Journal of Network and computer Applications*, vol. 35, no. 3, pp. 867–880, 2012.

[13] K. Zhang, X. Liang, R. Lu, and X. Shen, “Sybil attacks and their defenses in the internet of things,” *IEEE Internet of Things Journal*, vol. 1, no. 5, pp. 372–383, 2014.

[14] R. Gunturu, “Survey of sybil attacks in social networks,” *arXiv preprint arXiv:1504.05522*, 2015.

[15] C. Piro, C. Shields, and B. N. Levine, “Detecting the sybil attack in mobile ad hoc networks,” in *Securecomm and Workshops, 2006*. IEEE, 2006, pp. 1–11.

[16] P. Winter, R. Ensafi, K. Loesing, and N. Feamster, “Identifying and characterizing sybils in the tor network,” *arXiv preprint arXiv:1602.07787*, 2016.

[17] D. Gambetta *et al.*, “Can we trust trust,” *Trust: Making and breaking cooperative relations*, vol. 13, pp. 213–237, 2000.

[18] A. Jøsang, R. Ismail, and C. Boyd, “A survey of trust and reputation systems for online service provision,” *Decision support systems*, vol. 43, no. 2, pp. 618–644, 2007.

[19] T. Eder, D. Nachtmann, and D. Schreckling, “Trust and reputation in the internet of things,” Tech. Rep., 2013. [Online]. Available: https://web.sec.uni-passau.de/projects/compose/papers/Eder_Nachtmann_Trust_and_Reputation_in_the_Internet_of_Things.pdf

[20] F. Bao and I.-R. Chen, “Dynamic trust management for internet of things applications,” in *Proceedings of the 2012 international workshop on Self-aware internet of things*. ACM, 2012, pp. 1–6.

[21] M. K. Devi, R. T. Samy, S. V. Kumar, and P. Venkatesh, “Probabilistic neural network approach to alleviate sparsity and cold start problems in collaborative recommender systems,” in *Proceedings of The 2010 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*. IEEE, 2010, pp. 1–4.

[22] F. Isinkaye, Y. Folajimi, and B. Ojokoh, “Recommendation systems: Principles, methods and evaluation,” *Egyptian Informatics Journal*, vol. 16, no. 3, pp. 261–273, 2015.

[23] R. Azmi, M. Hakimi, and Z. Bahmani, “Dynamic reputation based trust management using neural network approach,” *International Journal of Computer Science Issues (IJCSI)*, vol. 8, no. 1, pp. 161–165, 2011.

[24] D. F. Specht, “Probabilistic neural networks,” *Neural networks*, vol. 3, no. 1, pp. 109–118, 1990.

[25] L. W. Steenhoek, M. K. Misra, W. D. Batchelor, and J. L. Davidson, “Probabilistic neural networks for segmentation of features in corn kernel images,” *Applied engineering in agriculture*, vol. 17, no. 2, p. 225, 2001.

[26] D. Grau and N. Sereni. Parallel computing for neural networks. Last accessed December 17, 2017. [Online]. Available: http://meseec.ce.rit.edu/756-projects/spring2013/1-4.pdf

[27] H. Tian, Z. Chen, C.-C. Chang, M. Kuribayashi, Y. Huang, Y. Cai, Y. Chen, and T. Wang, “Enabling public auditability for operation behaviors in cloud storage,” *Soft Computing*, vol. 21, no. 8, pp. 2175–2187, 2017.

[28] A. M. Antonopoulos, *Mastering Bitcoin: unlocking digital cryptocurrencies.* " O’Reilly Media, Inc.", 2014.

[29] Hyperledger fabric documentation. Accessed on April 28, 2017. [Online]. Available: https://hyperledger-fabric.readthedocs.io

[30] C. Cachin, “Architecture of the hyperledger blockchain fabric,” IBM, Tech. Rep. CH-8803, 2016. [Online]. Available: https://www.zurich.ibm.com/dccl/papers/cachin_dccl.pdf

[31] A. Moinet, B. Darties, and J.-L. Baril, “Blockchain based trust & authentication for decentralized sensor networks,” *arXiv preprint arXiv:1706.01730*, 2017.

[32] Hyperledger fabric composer participants and identities. Accessed on October 15, 2017. [Online]. Available: https://hyperledger.github.io/composer/managing/participantsandidentities.html

[33] N. Aberomand, “Network intrusion detection classification using optimized probabilistic neural network,” in *Proceedings of The 3rd International Conference on Computer Supported Education (COSUE’15)*, 2015, pp. 108–110.

[34] D. Evangelista, F. Mezghani, M. Nogueira, and A. Santos, “Evaluation of sybil attack detection approaches in the internet of things content dissemination,” in *Proceedings of The 8th Wireless Days Conference.* IEEE, 2016, pp. 1–6.