

A SECURITY PROTOCOL FOR IOT NETWORKS USING BLACKLISTING AND TRUST SCORING

A Thesis

by

Cem Ata Baykara

Submitted to the
Graduate School of Sciences and Engineering
In Partial Fulfillment of the Requirements for
the Degree of

Master's Degree

in the
Department of Computer Science

Özyeğin University
June 2022

Copyright © 2022 by Cem Ata Baykara

A SECURITY PROTOCOL FOR IOT NETWORKS USING BLACKLISTING AND TRUST SCORING

Advisor: Asst. Prof. Kübra Kalkan

Co-Advisor: Dr. Ilgın Şafak

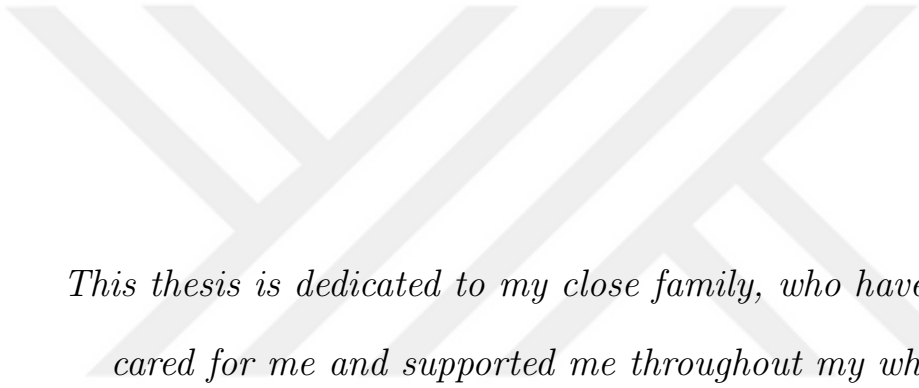
Approved by:

Asst. Prof. Kübra Kalkan, Advisor,
Department of Computer Science
Özyeğin University

Assoc. Prof. Hasan Sözer,
Department of Computer Science
Özyeğin University

Date Approved:

Prof. Fatih Alagöz,
Department of Computer Engineering
Boğaziçi University



*This thesis is dedicated to my close family, who have devotedly
cared for me and supported me throughout my whole life.*

ABSTRACT

There have been a number of high-profile incidents to compromise and attack larger networks of IoT devices, drawing attention to the need for IoT security. The purpose of IoT security is to ensure the availability, confidentiality, and integrity of IoT networks. However, due to the heterogeneity of IoT devices and the possibility of attacks from both inside and outside the network, securing an IoT network is a difficult task. Handshake protocols are useful for achieving mutual authentication which allows secure inclusion of devices into the network. However, they cannot prevent malicious network-based attacks once attackers enter the network. Use of autonomous anomaly detection and blacklisting prevent nodes with anomalous behavior from joining, re-joining, or remaining in the network. This is useful for securing an IoT network from insider network-based attacks. Similarly, trust scoring is another popular method that can be used to increase the resilience of the network against behavioral attacks.

The contributions of this thesis are threefold. First, we propose a new handshake protocol that can be used in device discovery and mutual authentication to ensure the security of the IoT network from outsider attacks. In the proposed handshake protocol, a Physical Unclonable Function (PUF) is utilized for the session key generation to reduce computational complexity. The proposed protocol is resilient to Man-in-the-middle, replay and reforge attacks as proven in our security analysis. Secondly, we propose a machine learning (ML) based intrusion and anomaly detection to prevent network-based attacks from the insiders. Finally, we propose a trust system which utilizes blockchain for managing the trust of a dynamic IoT network to increase resilience against behavioral attacks. Simulation results show that the proposed comprehensive security framework is capable of ensuring the security of an IoT network from both inside and outside attackers.

ÖZETÇE

Bilgisayar ağlarını tehlikeye atmak ve bunlara saldırmak için kullanılan IoT cihazlarının bir dizi yüksek profilli olayı IoT güvenliği ihtiyacına dikkat çekmektedir. IoT güvenliğinin amacı, IoT ağlarının kullanılabilirliğini, gizliliğini ve bütünlüğünü sağlamaktır. Ancak, IoT cihazlarının heterojen yapısı, IoT ağ güvenliğini sağlamayı zor bir iş haline getirmektedir. Güvenlik protokolleri, cihazların güvenli bir şekilde ağa dahil edilmesini ve karşılıklı kimlik doğrulaması sağlayan yararlı bir yöntemdir. Ancak saldırganların ağa katıldıktan sonra kötü amaçlı faaliyetlerini önlemek için yeterli değildir. Bu durumda ağ içerisinde anormal davranışa sahip cihazların ağa katılmasını, yeniden katılmasını veya ağda kalmasını engelleyen kara listeleme yararlı olabilecek bir yöntemdir. Benzer şekilde, güven puanlaması, ağ içerisindeki cihazların davranışlarını yönetmek için kullanılacak bir diğer popüler yöntemdir.

Bu çalışmanın katkıları üç yönlüdür. İlk olarak, IoT ağının dışarıdan gelen saldırılara karşı güvenliğini sağlamak için cihaz keşfinde ve karşılıklı kimlik doğrulamasında kullanılacak, hesaplama karmaşıklığını azaltmak için oturum anahtarı üretimi için Fiziksel Klonlanamayan Fonksiyon (FKF) kullanan, yeni bir el sıkışma protokolü öneriyoruz. Önerilen protokol, güvenlik analizimizde kanıtlandığı gibi, ortadaki adam, yeniden oynatma ve yeniden biçimlendirme saldırılarına karşı dayanıklıdır. İkinci olarak, içeriden gelen ağ tabanlı saldırıları önlemek için makine öğrenimi tabanlı bir saldırı ve anormallik tespiti öneriyoruz. Ve son olarak, dinamik bir IoT ağının güvenliğini yönetmek için bir güven yönetimi modeli öneriyoruz. Blok zinciri kullanan önerilen güven yönetimi modeli, hem istemcilerin hem de sunucuların güveninin belirsiz olduğu IoT ağlarında kullanılabilir. Simülasyon sonuçları, önerilen güvenlik çerçevesinin bir IoT ağının güvenliğini hem içeriden hem de dışarıdan gelen saldırılara karşı sağlayabildiğini göstermektedir.

ACKNOWLEDGEMENTS

I would like to thank my Advisors; Asst. Prof. Kübra Kalkan Çakmakçı and Dr. Ilgın Şafak. Their feedback and continued support and motivation have been invaluable to me not only academically but also in real life. I would also like to thank my close family members, my father Tefik Cem Baykara, my dear mother Ayşem Baykara and my grandmother Sabiha Mefharet Ceylan for their endless love and support, who made all my achievements possible and made me who I am today. Last but not least, I would like to thank my close friends Ozan Tarlan and Alp Demirezen, who always make me happy and are always there for me.

TABLE OF CONTENTS

DEDICATION	iii
ABSTRACT	iv
ÖZETÇE	v
ACKNOWLEDGEMENTS	vi
LIST OF TABLES	ix
LIST OF FIGURES	x
I INTRODUCTION	1
1.1 Motivation	1
1.2 Concepts	1
1.2.1 Constrained Application Protocol (CoAP)	2
1.2.2 Physical Unclonable Functions	2
1.2.3 Trust in Computer Networks	3
1.2.4 Blockchain	3
1.3 Contributions	4
II LITERATURE REVIEW	7
III SECURITY FRAMEWORK	14
3.1 System Model	14
3.2 Adversary Model	15
3.2.1 Protocol Based Attack Model	16
3.2.2 Network-Based Attack Model	16
3.2.3 Trust-Based Attack Model	17
3.3 Security Protocol	18
3.3.1 Registration	20
3.3.2 Connection	22
3.4 Security Analysis	24
3.4.1 Guarantee 1	25
3.4.2 Guarantee 2	26

3.4.3	Guarantee 3	27
3.4.4	Guarantee 4	27
3.5	Anomaly Detection and Blacklisting Model	28
3.6	Trust Management Model	29
IV	PERFORMANCE EVALUATION	32
4.1	Dataset Description	32
4.2	Performance Evaluation of the Classifiers	33
4.3	Network Simulation	35
4.4	Network Simulation of the Anomaly Detection Model	35
4.5	Network simulation of the Trust Model	39
4.5.1	Explicit Attack Scenario	40
4.5.2	Random Attack Scenario	41
4.5.3	Opportunistic Attack Scenario	42
4.5.4	Collusion Attack Scenario	43
4.5.5	Discussion of the Trust Model Simulation	44
V	CONCLUSIONS	46
	REFERENCES	48

LIST OF TABLES

1	Definitions and acronyms.	19
2	Label features of the IoTID20 dataset.	33
3	Macro Avg. Performances of the classifiers trained on category as target label.	34
4	Macro Avg. Performances of the classifiers trained on sub-category as target label.	34
5	Number of high-low complexity devices with their roles in the network simulation.	36
6	Performance of the CoAP server on detecting anomalous packets correctly.	37
7	Performance of the CoAP server on correctly classifying packets inside the network.	38

LIST OF FIGURES

1	The proposed network architecture.	14
2	Message sequence chart for the registration phase.	21
3	Message sequence chart for the connection phase.	23
4	F1 Scores of the classifiers on the test data for sub-category label. .	35
5	Performance of the trust provider against explicit attacks.	41
6	Performance of the trust provider against random attacks.	42
7	Performance of the trust provider against opportunistic attacks. . .	43
8	Performance of the trust provider against collusion attacks.	44



CHAPTER I

INTRODUCTION

Today IoT has become an integral part of our everyday life since it enables devices to be aware of their surroundings, efficiently communicate, and ultimately create a better networking environment [1]. While these smart devices offer tremendous opportunities, they also cause huge security challenges and risks [2]. These devices have differing computational complexities, and not all of them are equipped with means to mitigate malicious malware. Many IoT devices are manufactured with inherent security vulnerabilities [3]. Due to the increasing use of these vulnerable IoT devices to perform digital transactions, these vulnerabilities pose a risk to the security of an entire IoT network [4].

1.1 Motivation

The motivation of this thesis, which extends my previous paper [5], is to demonstrate that the security of an IoT network can be improved in performing digital transactions against both inside and outside attacks. However, ensuring security for all nodes with different computational complexities is a daunting task in a dynamic network where a significant number of nodes are constantly joining and leaving, and the trust of both the clients and servers is ambiguous. To protect the security of such a network, the approach needs to be lightweight on the IoT nodes so that even the devices with lower computational complexities can be allowed to join the network.

1.2 Concepts

This section briefly introduces the main concepts used within this thesis.

1.2.1 Constrained Application Protocol (CoAP)

CoAP is a specialized internet application layer protocol that allows constrained nodes to communicate with the Internet or with each other in a lightweight way suitable for IoT devices [6, 7, 8, 9]. CoAP provides a request, and response-based interaction model between nodes, while supporting built-in discovery of services and resources. The key features of CoAP that led us to implement the proposed security protocol on top of it includes, but are not limited to [9]:

1. Web protocol fulfilling machine-to-machine (M2M) requirements in constrained environments.
2. Asynchronous message exchanges.
3. Low header overhead and parsing complexity.

1.2.2 Physical Unclonable Functions

Physical Unclonable Function (PUF) is a physical object that, for a given input (challenge), produces an unclonable output (response) that serves as a unique identifier for that specific object [10]. The response generated by the PUF can also be called the digital fingerprint of that object. This concept is often achieved by a semiconductor device such as a microprocessor.

The inimitable feature gives PUFs an advantage over other hardware-based security concepts as even if the attacker has physical access to the device, they cannot clone its intrinsic properties [10]. Thus, PUFs enable us to perform device identification and authentication in a secure manner. It is also stated in [10] that PUFs provide a low-cost alternative when compared with the conventional methods for cryptographic key generation, making them a suitable solution for low complexity IoT devices. The security protocol presented in this paper utilizes PUFs to generate the secure session key.

1.2.3 Trust in Computer Networks

Due to the ever-growing size of computer networks and systems, the issue of trust has become an important issue [11]. Trust is an extremely complex concept and therefore it is important to describe its meaning as used in this study. Trust of entities inside a computer network is usually assigned based on past behavior or experience [12]. The main purpose of using trust scoring is to predict the behavior of entities in the network by looking at their past behavior. To accurately determine trust scoring, evidence and information about the interactions and behavior of entities must be reliable and secure. More information and evidence usually lead to better predictions, also called as trust scores [13].

Trust score can be seen as providing information to an entity, basically making the trust system a valuable service. In a computer network setting, even though there is nothing wrong with entities inside the network calculating the trust scores themselves, [13] claims that it is more efficient to rely on a service provider. The trust management model proposed in this thesis adopts this proposition to rely on trusted third parties, called trust providers, for the transformation of past behavior and transaction data into trust scores.

Trust systems are also subject to trust-based attacks, which make trust calculations misleading or inconsistent [14]. Therefore, additional measures to increase the robustness of the trust systems against such attacks are needed.

1.2.4 Blockchain

Blockchain is a distributed digital ledger technology that holds information about transactions and other types of data, accessible from the entire network [15, 16]. Properties of the blockchain technology such as verifiability (transactions on the blocks can be linked back to the parties involved) and immutability (it is extremely difficult to alter the transaction data once they are recorded in a block) are crucial. Furthermore, once a transaction data is recorded in a block, it immediately becomes transparent and universal. By storing information about interactions

and past behavior on the blockchain, it becomes accessible to all trust providers in the network, allowing them to base their trust score calculations on the same information, even if some use different calculation algorithms. [13] argues that the use of blockchain is an important contribution in this regard, which will allow distributed trust systems to be more robust and overcome many challenges such as centralization.

1.3 Contributions

The contributions of this thesis are three-fold. Firstly, we propose a security protocol suitable for detection of IoT devices of any computational complexity for the purpose of device discovery, device management and secure mutual authentication in an IoT network. A PUF is used in securely generating a session key by the IoT device; this is useful for secure authentication and reducing the computational load on IoT devices. Secondly, an autonomous real-time anomaly detection using ML, whitelisting and blacklisting approach is proposed to prevent unwanted or malicious devices from joining, re-joining, or remaining in the IoT network. Lastly, we propose a trust system which utilizes blockchain for managing the trust of a dynamic IoT network so as to increase resilience against behavioral attacks in central IoT networks where trust of both clients and servers is unknown. Simulation results show that the proposed trust management model is capable of assigning accurate trust scores in an IoT network with malicious nodes of varying behaviors.

IoT network simulation results are provided of the proposed security framework. After the devices enter the network using the proposed handshake protocol, the CoAP server in the network simulation performs real-time autonomous anomaly detection using pre-trained ML algorithms. IoT nodes within the network perform transactions through the server are assigned with trust scores by the trust provider based on their behavior. The framework ensures privacy against protocol-based outsider attacks through the use of the proposed security protocol, whereas blacklisting using autonomous anomaly detection and trust management

model protects the system from network- and behavior-based insider attacks.

To the best of our knowledge, the security framework proposed in this thesis is the first to address both internal and external security challenges of IoT networks, providing security against protocol-, network-, and behavior-based attacks using PUFs, autonomous whitelisting/blacklisting and trust systems. The contributions of this thesis are as follows.

1. The proposed protocol allows central authorities of the IoT network to establish negotiated communications, provide novel services, and autonomously prevent undesired devices from joining, re-joining, or remaining in the IoT network using whitelisting and blacklisting.
2. The proposed protocol allows central authorities of the IoT network to establish negotiated communications, provide novel services, and autonomously prevent undesired devices from joining, re-joining, or remaining in the IoT network using whitelisting and blacklisting.
3. Proposed security protocol, implemented on top of CoAP, allows device discovery, whitelisting, and mutual authentication for IoT devices. The protocol utilizes PUFs for both authentication and reduces computational load.
4. This thesis proves that the proposed protocol, and its authentication method, are secure against our threat model, including Man in the Middle (MITM), forgery and replay-based attacks.
5. A ML-based, autonomous, real-time anomaly detection system that uses blacklisting is proposed to secure the IoT network from insider network-based attacks.
6. A blockchain-based trust management model is proposed, where a trusted third party, called the trust provider, manages the trust scores of both clients and servers in the network, based on the evidence of their verifiable interactions to increase the resilience of the system against behavior based attacks.

7. The thesis presents an IoT network simulation to test the effectiveness of the proposed security protocol, anomaly detection system and the trust management model. The simulation includes a CoAP server that allows connection via the proposed protocol and performs real-time anomaly detection using pre-trained ML classifiers. The thesis also presents detailed discussion of the relative merits of the considered ML classifiers. Similarly, the trust provider performs real-time trust management and assignment of trust scores to the IoT devices in the network to mitigate behavior-based attacks.
8. Based on the analysis and network simulations performed, the security framework proposed in this paper can easily be adapted to real-life IoT networks.

The remainder of this thesis is structured as follows. Chapter II presents current approaches and literature review. Chapter III presents our system and adversary model, along with a detailed description of the proposed framework, which includes the handshake protocol and its security analysis, the anomaly detection model, and the trust management model. Chapter IV presents the performance analysis of the proposed security framework, namely the network simulation and its results. Finally, Chapter VI presents the conclusions of this thesis.

CHAPTER II

LITERATURE REVIEW

This chapter provides a summary of related work and literature review and discusses the differences between them and the works proposed in this thesis.

In the Transport Layer Security (TLS) protocol, session keys are randomly generated by the client [17]. This approach may be prone to replay and forgery attacks, depending on how the client generates the random string. Compared to the TLS protocol, the proposed security protocol in this thesis is more lightweight and secure to mutual authentication, since it performs mutual authentication by generating session keys using a PUF and utilizing a whitelisting and blacklisting approach in order to autonomously ensure IoT network security using ML.

An approach to using PUFs to decrease the computational power in IoT devices is proposed in [18]. This work shows in detail the benefit of using PUFs to establish a secure session with IoT devices by comparing their proposed protocol with other existing solutions such as Datagram Transport Layer Security (DTLS) handshake protocol and User Datagram Protocol (UDP). The results indicate that, with the use of PUFs, the authentication process achieves a power reduction of up to 45% and requires 12% less memory, compared with the existing solutions listed before. The authors propose a method of detecting artificially generated challenge-response pairs (CRPs) by using neural networks that their proposed verifier (server) uses to authenticate the nodes. By employing this technique, the server can detect malicious nodes that try to authenticate with the server by using replayed CRPs. However, the protocol can only authenticate existing and trusted nodes in the network; it does not utilize whitelisting or ML techniques for autonomous blacklisting of suspicious devices as proposed in this paper. Thus, it is not suitable for general and public use since it constrains the system from allowing

new nodes to join the network. In contrast, the protocol presented in this thesis, while keeping a record of previously explored nodes, allows new nodes to join the network and uses whitelisting and blacklisting techniques in ensuring the security of the IoT network.

Lightweight key exchange protocols that use pre-shared secret symmetric keys are proposed in [19, 20]. These protocols assume that one or more symmetric keys are readily available to the client and server before the handshake protocol begins. However, this is not a secure or scalable approach since it requires every new client to obtain the shared keys a forehand and is prone to forgery and brute force attacks. In comparison, the proposed handshake protocol in this thesis utilizes challenge response pairs to generate a secure session key and achieve mutual authentication on-the-fly without requiring pre-shared symmetric keys, making it more secure against reforge and brute force attacks, as well as utilizing whitelisting and blacklisting in ensuring network security.

Lightweight key exchange protocols using mutual authentication are proposed in [21, 22]. These protocols use PUFs not only for generating session keys, but also for registration and authentication purposes. However, these protocols do not study autonomous blacklisting techniques in ensuring network security. The protocols store long-term keys to authenticate the server or a device, a process which requires extensive computation. Given the limited computational capability of IoT devices, the proposed protocol in this paper tries to minimize the computation required from the IoT nodes for server authentication by authenticating the server with the help of a Certificate Authority (CA). However, using CA has its downsides as it requires the assumption that the CA will always be available since the protocol cannot be completed otherwise. Using CA for authentication also increases the communication flow required to perform the handshake. The proposed protocol in this paper performs the key exchange with 7 communication steps, whereas the protocol proposed in [22] only needs 3 communication steps for a registered user. However, unlike these PUF-based authentication and key

exchange protocols, the protocol presented in this paper lets any user initiate the key exchange without client registration or client verification processes.

Similar handshake protocols are proposed in [23]. These protocols also utilize the low computational complexity of PUFs for IoT devices and thus minimize the requirements for computation, memory, and communication overhead. The results in [23] show that using PUFs for secure session generation is a suitable solution for IoT devices. Another protocol proposed in [24] can generate a secure session between IoT devices. This protocol uses a trusted and pre-authenticated server for secure session generation, similar to the handshake protocol proposed in this paper, which uses a CA for server authentication. Results in [24] are also in agreement with the previous related work and show the effectiveness of using PUFs instead of the existing solutions. However, even though authors of [23] state that their proposed protocols can be adapted to real-life use cases, they do not provide a real experimental setup or a real time simulation to prove this statement. In contrast, the results and details of a real time network simulation show that the handshake protocol proposed in [24] can easily be adapted to real life use cases.

[25] proposes a secure PUF-based authentication and identity-based key exchange protocol suitable for distributed IoT networks. It differs from this thesis in that a certificate-less identity based key exchange approach is used, whitelisting and blacklisting is not incorporated, and CRPs are not used to establish mutual authentication.

[26] proposes a lightweight mutual authentication protocol based on a new public key encryption scheme based on transmitting challenges and checking whether the recipient can respond accordingly. This is similar to the use of CRPs in the protocol proposed in this thesis. The protocol is shown to have a performance significantly better than existing RSA- and ECC-based protocols. In contrast with the protocol proposed in this thesis, this protocol does not require the use of a CA for authentication, does not include whitelisting and blacklisting using ML techniques and real-time trust-scoring based on blockchain in order to ensure the

network security.

[27, 28] propose variations of the Datagram Transport Layer Security (DTLS) handshake protocols for IoT networks. [27] proposes a simplified version of the Datagram Transport Layer Security (DTLS) handshake protocol suitable for IoT devices for a general scenario of end-to-end communications based on software-defined networking (SDN). A controller is utilized in generating a symmetric key dynamically, then encrypting and distributing the key to two communicating IoT devices. Certificate verification is shifted from the IoT device to the more powerful controller, where the controller replaces the DTLS server to make a cookie exchange with the DTLS client. The computational overhead, the energy consumption in the IoT devices and the overall duration of the handshake protocol are shown to reduce. [28] separates the DTLS protocol into the handshake phase and the encryption phase, which is shown to enhance the performance of both the device and the network by using a way to delegate the DTLS handshake phase. The proposed scheme supports secure end-to-end communication despite using delegation. However, neither paper utilizes whitelisting or blacklisting methods in ensuring the network security and trust scoring as proposed in this thesis.

[29] proposes a different fingerprinting approach to keep a log of previously known clients and detect malware and other malicious processes trying to initiate a secure connection using the TLS handshake protocol before the secure session is established. The fingerprinting technique proposed in [29] uses the initial unencrypted hello message sent by the clients after the TCP connection is established. Since this message is unencrypted, TLS fingerprinting extracts metadata presented in the message and generates a fingerprint string using a predefined scheme. After generating a fingerprint, the server maps it to the client by keeping a dictionary of known fingerprints to client mappings. By doing so, the server can detect previously known or blacklisted clients trying to initiate a connection and take actions accordingly. However, it is stated that the TLS fingerprinting technique is not sufficient per se to profile clients effectively. A single fingerprint may map to tens

or hundreds of unique clients. Thus, the TLS fingerprinting itself is often a poor indicator and additional information is required to increase its reliability [29]. The handshake protocol presented in this thesis uses PUFs and CRPs for both client fingerprinting and session key generation. In order to verify the identity of a client, the server may log the challenges and the session keys used to create sessions and authenticate clients by sending the logged challenges. Since the clients in the proposed handshake protocol use their PUFs to generate the session keys, the client is expected to generate the same session key given the same challenge where only a specific client can create that unique session key due to how PUFs work. This approach is both more precise compared to [29], and it does not require additional information about the client for fingerprinting. Hence, the handshake protocol proposed in this thesis is more lightweight than TLS fingerprinting technique in [29] which puts additional computational overhead on the server.

A trust management framework based on verifiable interactions is proposed in [13] which uses both blockchain and trust providers, similar to the trust management model proposed in this thesis. Blockchain is utilized to create the evidence on which trust providers base their trust score assignments for universal, transparent and verifiable trust scoring. [13] lays down in-depth feedback and evidence-based trust scoring through the use of evidence mapping. Proof-of-Interaction (PoI) is used instead of traditional Proof-of-Work (PoW), which is more suitable for low complexity IoT devices due to PoW requiring more computational power. [13] also discusses different trust score calculation formulas that can be used by the trust providers in such a system model. However, in-depth simulation results which include malicious devices with varying behaviors are not provided and only briefly touched upon. Moreover, the inclusion of devices into the network is not well studied as opposed to the model proposed in this thesis which uses a novel lightweight security protocol utilizing real-time whitelisting and blacklisting.

A framework for modeling trust and reputation systems in both peer-to-peer

(p2p) and central server-client (central) networks which include malicious participants is proposed in [30] called TRIM (Trust and Reputation Interaction Model). TRIM is proposed as an improved version of traditional TRS (Trust and Reputation) systems which the authors claim to be vulnerable in a hostile environment where trust-based attacks cause the TRS system to deteriorate. Authors also underline the lack of formal analysis on the robustness of TRSs against trust attacks due to their generality, expressiveness, or applicability. The TRIM framework they propose mainly involves modeling the interactions between the system's agents. The system uses two different models for interaction (primitive and compact) with various degrees of abstraction. Finally, the model specifies a TRS model and combines it with the interacting agents model to complete the TRIM structure which is capable of modelling trust attacks as well as other TRSs. The authors show that the proposed framework is a finite state non-homogeneous Markov decision process (MDP), which is transformed to an infinite state homogeneous MDP tree through their proposed algorithm. Finally, they propose a new robustness criterion is provided, based on the cost/reward mechanism, by solving the MDP tree of the TRIM. The comparison between TRIM and other traditional TRSs show that TRIM is more complete than the previous frameworks and is capable of modelling a wide range of TRSs in hostile networks. However, secure inclusion of the agents into the environment and protocol/network-based attack scenarios are not studied. This thesis utilizes a security protocol for secure authentication and inclusion of the devices into the network which is shown to be robust against various protocol based attacks, and blacklisting utilizing ML algorithms against network based attacks.

A trust management model for cloud computing systems is proposed in [31]. The model is capable of dynamically updating the trust values for each entity in the network and uses the concepts of experience and temporal reputation to calculate the direct trust of service providers in the cloud environment. However, the proposed model lacks application and only a theoretical case study is presented to

demonstrate its effectiveness. Also, adversary model and possible attack scenarios are not presented.

A trust model for distributed IoT networks is proposed in [32]. Authors claim that their proposed model, called CBSTM-IoT (Context-based Social Trust Model), is capable of establishing improved cooperation with trusted nodes and reducing the number of transactions with malicious nodes. The model proposed calculates trust based on social relationships between IoT nodes and the context of their interactions. The paper also presents an adversary model to show the resilience of CBSTM-IoT against various trust-based attack scenarios. The limitations of CBSTM-IoT include the computation of trust by each node in the system, which may not be suitable for lower complexity IoT devices. In fact, the authors state that since the trust of entities inside the system is individually calculated by each node, lower complexity devices require these calculations from other nodes, which introduces further security concerns in an environment with malicious nodes. The trust model proposed in this thesis uses trust providers to manage and calculate trust within the system, while it also allows nodes to calculate the trust of entities within the network themselves as the information used in trust calculations is publicly accessible via the blockchain.

CHAPTER III

SECURITY FRAMEWORK

This chapter describes the proposed framework in detail. The system architecture and adversary model is presented. Firstly, the proposed security protocol is presented along with its security analysis. Secondly, the autonomous anomaly detection model using ML algorithms is presented. And lastly, the trust management model is presented in detail.

3.1 System Model

The proposed network architecture is shown in Figure 1.

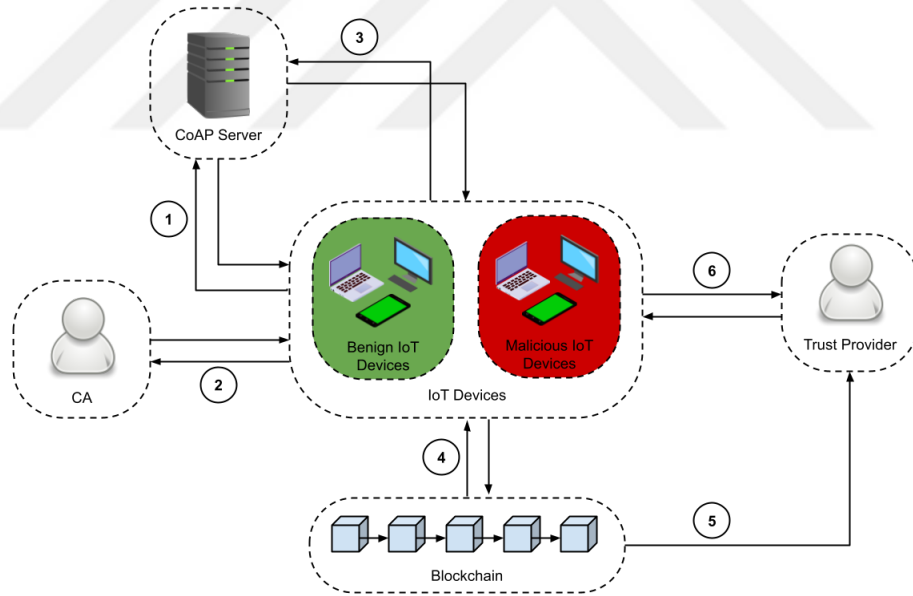


Figure 1: The proposed network architecture.

The network consists of a central CoAP server, a certificate authority (CA), heterogeneous IoT devices, blockchain, and a trust provider. The system flow is as follows:

- (1) The CoAP server establishes a secure connection with the IoT devices via

the proposed security protocol. The protocol has two phases, namely registration and connection.

(2) During both phases of the protocol, the IoT devices authenticate the identity of the CoAP server through a CA.

(3) After the devices establish a secure connection with the server using the proposed protocol, they start sending requests to the server. The CoAP server performs blacklisting inside the network using autonomous real-time anomaly detection.

(4) After each request, IoT devices score their interaction with the server and record it on the blockchain in the form of feedback. Every feedback is recorded with its relevant evidence. Evidence of an interaction may include any service-related feature provided through an interaction. Since each device inside the network has access to the blockchain, all the feedbacks and their evidence is universal and transparent for all participants inside the network.

(5) The trust provider uses the feedback and evidence information stored in blockchain to calculate the trust of entities inside the network. The trust of both server and the IoT devices are calculated by the trust provider.

(6) The CoAP server and IoT Devices within the network request the calculated trust scores from the trust provider. Depending on the trust scores provided, devices can decide to stay within the network or leave the network, while the server may blacklist devices with low trust scores.

3.2 Adversary Model

The adversary model assumed in this thesis includes possible attacks from both outside and inside the network. The three main forms of possible attacks considered for the proposed system model are protocol-based attacks, network-based attacks, and behavior-based attacks. This section details the default attacker models for each attack type in detail.

3.2.1 Protocol Based Attack Model

The Dolev-Yao attacker model provided in [33] is used. The attacker can listen, replay, and create messages in the network:

- To generate or obtain the keys used in the session.
- To acquire device information.
- To obtain the confidential information shared within the communication.
- To acquire the Challenge Response Pairs (CRPs) generated by the IoT nodes.

The network aims to provide a secure connection to any device that successfully performs both phases of the protocol. Therefore, the denial of service and jamming attacks are out of the scope for protocol-based attack model.

3.2.2 Network-Based Attack Model

After devices establish a secure connection with the server, some devices may generate malicious network traffic within the network. The network based attacks considered in this thesis are as follows:

- Host Brute Force: A type of attack using Mirai malware that largely infects IoT devices. There are many sub-varieties of Mirai malware, but the core concept of Mirai malware is to take control of an IoT device by brute-forcing using dictionary-based techniques to guess passwords [34].
- HTTP Flooding: HTTP GET flooding is one of the most effective DDoS attacks. It is performed by flooding the target server with a large number of HTTP requests aimed at slowing down or completely disrupting normal network traffic [35, 36].
- UDP Flooding: One of the most commonly used forms of DoS (Denial of Service) and DDoS (Distributed Denial of Service) attacks. Its limited ability to respond to requests and the connectionless property of UDP are exploited

in the UDP flood attack. It is performed by submitting a stream of UDP data packets to the target to fill its responding request queue and stop benign users from accessing the target [37].

- **Host Port Scan:** Port scan attack is performed by sending a TCP or a UDP probe packet to ports of the target host to gain information on which ports are actively listened by the target. Discovered ports may then be used for other attacks [38].
- **Host OS Scan:** An attack similar to a port scan attack aimed to obtain information about which operating system (OS) the target machine is running on.

Since the communication of the devices within the network is secured using the proposed handshake protocol, Man-in-the-Middle (MITM) attacks are out of scope for the network-based attack model.

3.2.3 Trust-Based Attack Model

Malicious nodes inside the trust system may carry out various attacks to make the trust score calculations inconsistent or misleading. Since only a single trust provider is used in the proposed system architecture of this thesis, the trust provider is always assumed to be non-malicious.

The attacks on the trust system considered for this thesis may be classified as follows:

- **Explicit Attacks:** These attacks include nodes which always give inconsistent feedback in an attempt to bad mouth a legitimate server. These nodes do not actively try to avoid detection and do not care about their trust in the system.
- **Random Attacks:** These attacks include nodes which behave randomly when giving feedback to avoid detection.

- **Opportunistic Attacks:** These attacks include nodes which behave as benign nodes at first, giving consistent feedback based on the evidence of their interactions in an attempt to increase their trust score. After achieving a certain amount of trust score, or after giving a certain amount of legitimate feedback, they start acting maliciously, such as performing good or bad mouthing.
- **Collusion Attacks:** These attacks include nodes which act in groups. They pretend to be benign nodes by giving consistent feedback, then suddenly start good or bad mouthing together.

3.3 Security Protocol

This section presents both phases of the proposed protocol and its security analysis in detail. The protocol allows IoT devices to establish a secure session for communication by utilizing asymmetric encryption, PUFs and Challenge Response Pairs (CRPs). Clients and servers that follow the protocol can securely generate a one-time session key to achieve end-to-end encryption for secure communication. Definition of acronyms used within this section are provided in Table 1, and the message sequence charts of each phase are shown in Figures 2 and 3.

The proposed protocol does not assume initially trusted devices and therefore all devices are required to perform both phases of the protocol to establish a secure connection with the server. The CA is utilized in both phases to authenticate the server. This verification process is assumed to be secure and encrypted between the device and the CA and its security is not within the scope of this thesis.

Using PUFs and CA for key generation and server authentication respectively provides efficiency and consistency for IoT devices, which is crucial for use cases that require both communicational integrity and low computational power such as secure payment. The server adds to the IoT network the devices that successfully complete both phases of the protocol. The blacklist used within the protocol consists of devices detected using the anomaly detection model. Anomalous network

traffic, which may pose a threat to the network, is detected by the server and the devices that produce the anomalous traffic are removed from the IoT network and recorded into the blacklist. Further network traffic from the devices within the blacklist is declined by the server. The proposed protocol provides a secure and lightweight solution for IoT device communication on CoAP. The key exchange message sizes are kept within the boundaries of the CoAP protocol.

Table 1: Definitions and acronyms.

<i>Acronym</i>	<i>Definition</i>
CA	Certificate Authority
r_C	Random nonce of client
r_S	Random nonce of server
$\oplus$	Exclusive OR operator
$\vee$	OR operator
$\parallel$	Concatenation operation
ch_S	challenge of server
$+K_{Server}$	Public key of the server
$-K_{Server}$	Private key of the server
$Certificate_{-CK_{CA}(+K_{Server})}$	Certificate of $+K_{Server}$ issued by CA
$CA \ll Server \gg$	Shorthand notation of $Certificate_{-CK_{CA}(+K_{Server})}$
$E_K(.)$	Encryption using key K
$D_K(.)$	Decryption using key K
$\theta(.)$	Physical Unclonable Function
$K_{session}$	Session Key
$S \xleftarrow{R} S_r$	Random re-ordering of S
$x \in_R S$	Randomly selecting an element from S
C	Challenge set
R	Response set
$\in$	Inclusion
ID_{Client}	Client ID
B	Blacklist
W	Whitelist
$ID_{session}$	session ID
$resp$	Response
I_C	Specifications of the Client
$h(.)$	One-way hash function

3.3.1 Registration

The registration phase is initiated by the client with a hello message. This first message includes a one-time use nonce called r_C . After receiving the hello message, the server generates its own nonce r_S and also calculates a challenge $r_C \oplus r_S = ch_S$. The server then replies to the client with a message including the signed certificate of the server issued by CA, r_S , and ch_S . After receiving this message, the client performs a check to see if $ch_S = r_C \oplus r_S$, to make sure that the server actually used the r_C when generating the challenge. If this check fails, the client terminates the connection immediately. After the client verifies that the r_C is used to obtain the ch , it sends the certificate of the server to the CA for authentication on message 3. This message is encrypted with the public key of the CA, and it also includes second nonce r_{C_1} generated by the client to show that the communication is live. If the certificate cannot be verified, the CA responds with a *NOK* message, in which case the client terminates the connection as the server cannot be trusted. If the client receives an *OK* message, it then uses its PUF to generate a one-time use session key as $\theta(ch_S) = K_{session}$. Message 5 is encrypted with the $+K_{server}$ and it includes the generated $K_{session}$ alongside the ch_S to prevent replay attacks. The server then decrypts the 5th message using its private key $-K_{server}$. The server keeps a challenge set C (which can be updated between pre-defined time intervals) to authenticate the IoT devices. Since any device, malicious or benign, can acquire the challenges in the C by initiating the registration phase, a way to prevent the attackers from mapping the challenges to their responses generated by IoT devices is required to prevent replay and forgery attacks. The server achieves this by randomizing the order of the challenges every time an IoT device initiates the registration phase as $C \xrightarrow{R} C_r$. After creating a randomly ordered C_r , the server symmetrically encrypts it with $K_{session}$ and sends it to client on message 6. This message also includes the initially generated ch_S to prevent replay attacks.

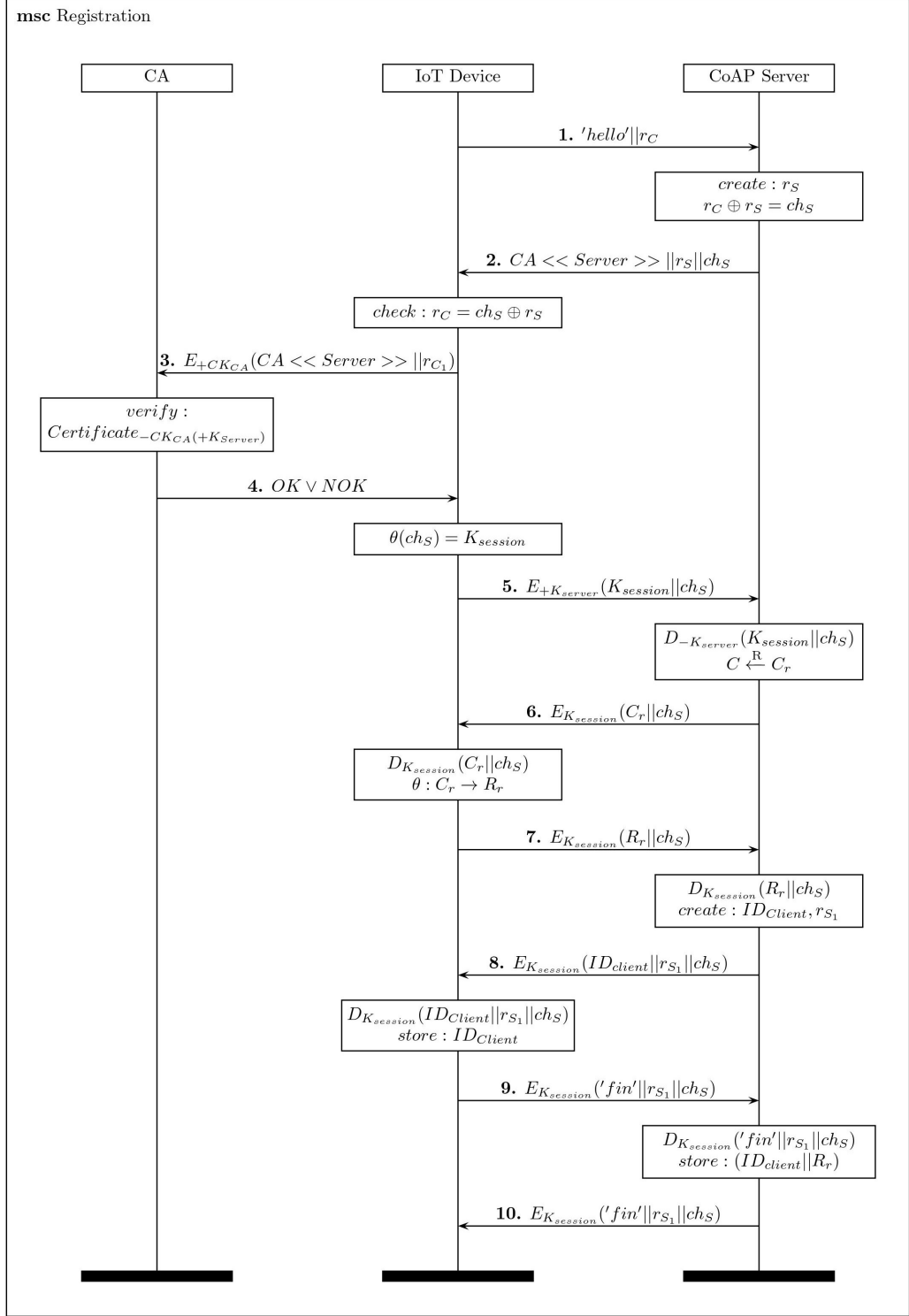


Figure 2: Message sequence chart for the registration phase.

After decrypting message 6 using $K_{session}$ and obtaining the randomized C_r , the client uses its PUF to generate a response set R_r for each challenge in the order they appear on C_r as $\theta : C_r \rightarrow R_r$. The client then encrypts the R_r using

the $K_{session}$ and sends it to the server on message 7. After receiving the response set sent by the client the server creates a unique ID for the client and a second nonce r_{S_1} . It then sends these back to the client on message 8. The server does not store R_r and ID_{client} yet because the registration phase is still not complete and the client may still drop out of the protocol. Only after the client sends a finished response on message 9 that the server stores ID_{Client} and R_r . The server completes the registration phase by sending the final finished message on the final message 10.

The registration phase is important in two respects. Firstly, it allows the IoT client to obtain a unique ID that can be used by the server for identification. Secondly, it allows the server to log the responses generated by the client's PUF for the challenge set, which can be used for authentication at the connection phase.

3.3.2 Connection

Connection phase is also initiated by the client with a hello message. Clients present their unique ID_{Client} in this hello message. This implies that all clients must complete the registration phase before attempting to initiate a connection. Alongside the ID_{Client} this message also includes a random nonce r_C . After receiving the hello message, the server checks its validity and checks if it has been blacklisted before. If the ID_{Client} passes both checks, the server then creates a random nonce r_S to calculate a challenge as $r_c \oplus r_S = ch_S$, and also randomly selects a challenge from C as $C \in_R c_r$. The server then sends a message back to the client which includes its certificate issued by the CA, c_r , r_S , and ch_S . Sending r_r unencrypted poses no security concerns as all the challenges in CP can be obtained by any client through the registration phase. Similar to the registration phase, the client authenticates the server through the CA by sending it the certificate of the server on message 3. The CA responds to the client with an *OK* or *NOK* message. If the certificate of the server can be verified and the client receives an *OK* message from the CA, then the client uses its PUF to generate both the session key and the response $resp_r$.

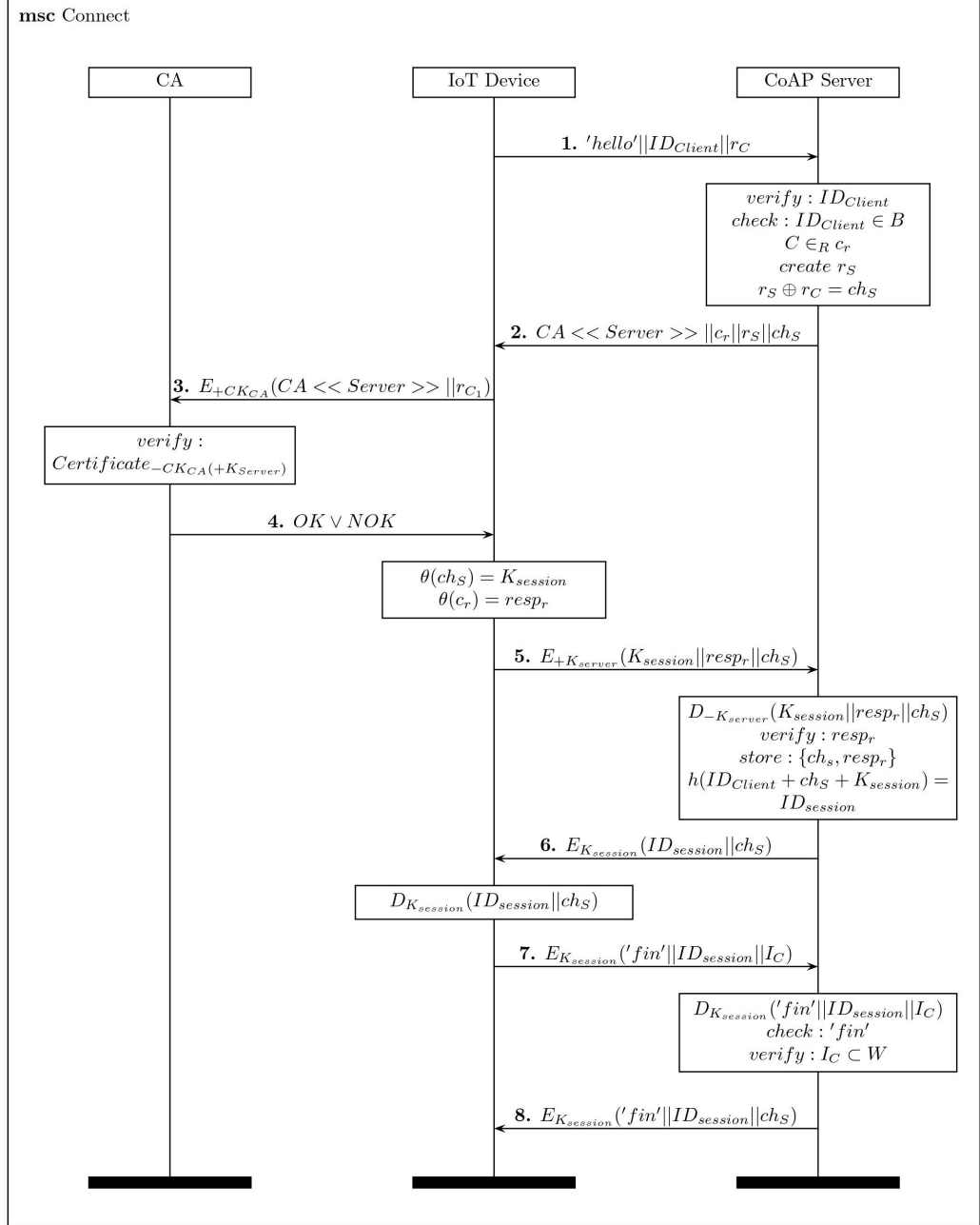


Figure 3: Message sequence chart for the connection phase.

The response generated will be the same as the response generated during the registration phase due to the properties of PUF, which allows the server to authenticate the client. After generating the response, the client encrypts it with $+K_{server}$ and sends it to the server on message 5 alongside $K_{session}$ and ch_S . The server then decrypts the message using its $-K_{server}$ and checks the response of the client against the response generated during the registration phase. A same

response means that the client can be successfully authenticated while a different response may indicate that the client may be trying to use the ID_{Client} of another client. If the responses do not match, the connection is terminated immediately by the server. Multiple failed connection attempts from the same devices can be subjected to a timeout to prevent server overload. If the responses match, the server stores ch_S and $K_{session}$ together, to be used at a later time to authenticate the same device. It also creates a unique session ID for the secure session that will be generated shortly using a one-way hash function $h(ID_{Client} + ch_S + K_{session}) = ID_{session}$, which is sent to the client on the 6th message alongside ch_S and encrypted with the $K_{session}$. After receiving and decrypting this message, the client responds to the server with a finished message which also includes its specifications on message 7. The server then decrypts message 7 and checks the whitelist for the specifications of the device. If the device is eligible to join the network, the server sends a final finished message back to the client on the 8th message. At this point the secure symmetric encryption has been achieved by the client and the server and the ensuing session is continued using K_S as the symmetric key.

The IoT devices that have previously gone through the registration phase and obtained their client IDs are enabled to establish a secure connection with the server by authenticating themselves using their PUF. The secure connection also allows the server to obtain information about the specifications of the IoT devices joining the network for both whitelisting and device discovery purposes.

3.4 Security Analysis

Based on the model described, the security analysis of the protocol is provided in this section. The likelihood of an attacker breaking the security guarantees of the protocol to achieve various goals is examined. The analysis is based on the following assumptions:

1. The signature scheme used by the protocol participants (CoAP server, IoT

client, and CA) is secure (it is impossible for an attacker to forge the signature without the private key).

2. PUFs used by the IoT devices to generate responses are based on intrinsic qualities of their hardware and therefore cannot be manipulated and/or stolen.
3. The communication between the IoT devices and the CA is secure.
4. The challenge set C contains enough challenges so that it is impossible for an attacker to predict different possible sequence combinations.
5. Random nonces generated by both client and CoAP server are never picked twice.

Based on these assumptions, the proposed security protocol provides 4 guarantees.

3.4.1 Guarantee 1

For the server, the protocol guarantees authentication with the legitimate owner of any ID_{Client} in the connection phase.

Proof An attacker may try to authenticate using the ID_{Client} of another client since it is shared unencrypted on the first message of the connection phase. There are 2 options for an attacker to authenticate using the ID_{Client} of another client. The attacker should either be able to provide the same response $resp_r$ generated by the client to the randomly selected c_r or replay a message which includes the response of the client generated from the same c_r . Because the client is using a PUF, the attacker cannot generate the same client-generated response for r_r . By assumption 2, since PUF is based on intrinsic qualities of the hardware of the device, it cannot be replicated and manipulated. Therefore, unless the attacker has physical access to the device itself, it is impossible for him to generate the same response. The other option for the attacker to authenticate as another client is to replay an already generated $resp_r$ for c_r . To achieve this, the attacker is

required to capture and replay the 5th message on the connection phase which must include a response generated from the same c_r . Since c_r is sent unencrypted on the 2nd message of the connection phase, the attacker may wait for it to be selected again and then replay the same message to authenticate. This is not possible due to both assumptions 2 and 5. Since the 5th message includes $K_{session}$ which is generated using a PUF as $\theta(ch_S) = K_{session}$, and because the behaviors of PUFs cannot be predicted without having access to the device itself, the attacker can never produce the same $K_{session}$ even if the attacker knows the ch_S . Even if the attacker somehow had access to the $K_{session}$, a replay attack is still not possible because both, the 5th message also contains ch_S , and also $K_{session}$ is produced using ch_S , and assumption 5 states that nonces cannot be picked twice.

3.4.2 Guarantee 2

For clients, the protocol guarantees authentication with the original server.

Proof If an attacker creates a fake server in registration and/or connection phase, he must provide clients trying to connect to his server with a signed certificate in the 2nd message of both phases to proceed with the security protocol. Since a client confirms the identity of the server through the certificate authority, the two ways an attacker can pose as the real server is by creating or replaying an *OK* message on the 4th message of either phase after the client presents the attacker's certificate in the 3rd message. The attacker can never generate the 4th message for an illegitimate certificate because of assumption 1, which states that the signature scheme used by the protocol participants is secure and it is impossible to forge a signature without the private key. Thus, the attacker cannot forge an *OK* message unless he has access to the private key of the CA which is never shared. Another option to authenticate as a fake server is to replay an *OK* message, which is also not possible because assumption 3 states that the communication between the CA and the client is secured. The security of the communication between the CA and the client implied in this assumption also means that the connection is live. Therefore, an attacker can never replay an *OK* message as, after encryption, the

same *OK* message is never generated again.

3.4.3 Guarantee 3

The protocol guarantees the security and confidentiality of the session generated after the connection phase.

Proof The attacker needs to either forge or replay the 5th message of the connection phase to break the security and confidentiality of the session generated between the IoT client and the CoAP server. To forge the session key sent on message 7, the attacker would need the θ function used by the client. This is not possible without having physical access to the device because of assumption 2. It is also not possible for the attacker to replay the message containing $K_{session}$ because the message also contains $resp_r$ and ch_s . $resp_r$ cannot be generated or replayed by the attacker due to assumptions 2 and 5, and also ch_s cannot be replayed due to assumption 5.

3.4.4 Guarantee 4

The protocol guarantees that the pairing of the response set R_r generated by the client to the challenge set C_r , where $C \xleftarrow{R} C_r$, is only known by the client and the server.

Proof One way an attacker can obtain clients' challenge-response pairs is to pretend to be a legitimate server and give clients a C_r whose order is known by the attacker. This is not possible because of assumption 3 and reasons explained in the 2nd guarantee. Another way to obtain the challenge-response pairs is to produce the response set R_r for a given C_r , which is also not possible without physical access to the device itself because of assumption 2. Lastly, challenge-response pairs of a client can be obtained by breaking the encryption of both the 6th and 7th message of the registration phase, which is only possible by knowing the $K_{session}$. Both of these messages are required to be decrypted to access the challenge-response pairs because the number of possible sequence combinations of a challenge set C increases very fast as the number of challenges increases due

to combinatorial growth of possible sequences. Since assumption 4 states that C contains a sufficient number of challenges, it is impossible to obtain challenge-response pairs without decrypting both messages. $K_{session}$ can only be obtained by either forging it by knowing the ch_S or by decrypting the 5th message, which is the only message where it is shared. Because of assumption 2, $K_{session}$ cannot be produced by the attacker, even if the attacker knows the ch_S . Therefore, the only other way to obtain the $K_{session}$ is to decrypt the 5th message, which is also not possible because of assumption 1.

3.5 Anomaly Detection and Blacklisting Model

The proposed security protocol allows mutual authentication and communication privacy but does not prevent non-blacklisted devices from entering the network. Therefore, to prevent possible attacks within the network, a blacklisting approach using autonomous anomaly detection using ML is proposed. The anomaly detection is performed by the CoAP server, on the network traffic generated within the network after the devices establish a secure session using the proposed protocol. Devices generating abnormal or malicious network traffic are detected and blacklisted by the server. Blacklisted devices are prevented by the CoAP server from joining, re-joining, or remaining in the network. Instead of traditional intrusion and anomaly detection methods, an approach using ML classifiers is proposed in this thesis. Use of ML classifiers for intrusion and anomaly detection in network security has been researched extensively in the past years [39, 40]. Classification algorithms considered in this thesis include:

- Random Forest (RF)
- Decision Tree (DT)
- Stacking
- K-Nearest Neighbor (KNN)
- Gaussian Naïve Bayes (GNB)

Decision Tree, K-Nearest Neighbor and Gaussian Naïve Bayes classifiers are preferred due to their suitability for intrusion and anomaly detection [40, 41]. As for Random Forest and Stacking classifiers, their high performance on the IoTID20 [42] dataset used to perform the network simulation in this thesis was a decisive factor. More information about the dataset will be presented in Chapter IV.

3.6 Trust Management Model

The trust management model implemented in this thesis is similar to the framework proposed in [13]. In our proposed network system model, the CoAP server is responsible for securing IoT devices by performing real-time anomaly detection and blacklisting. However, the proposed network model does not initially assume reliable servers, and therefore the CoAP server itself may be malicious and cannot be initially trusted. Therefore, clients collectively rate their interaction with the server based on different parameters of that interaction. IoT devices record their feedback on the blockchain so that trust providers can calculate the trust score of the server. IoT devices then request the current trust score of the server by interacting with the trust provider directly. The proposed system model is flexible, allowing the devices to calculate the server's trust score themselves if they don't want to use the trust service provided by trust providers, as all feedback and historical interaction data is universally available through the blockchain. The use of blockchain is also beneficial in a system architecture where multiple trust providers are present as it allows them to base their calculations on the same feedback and evidence set [13].

Proposed system model for the trust management model consists of a CoAP server, a trust provider, the blockchain and multiple IoT devices. Devices connect to the CoAP server by going through the proposed security protocol. Devices that enter the network start sending requests to the server and, based on the features of this interaction, they leave feedback on the blockchain. Each feedback saved on blockchain also includes various features, the so-called evidence, of the interaction

between the server and the client. These features as a whole represent the quality of the service provided. This thesis assumes that the evidence linked to a feedback score cannot be manipulated. The trust providers use the evidence and feedback score recorded on the blockchain to calculate the trust of both the server and the client. The trust of the server is based on the evidence and the feedback recorded on the blockchain, whereas the trust of the individual clients is calculated based on the consistency of their feedback with their evidence. The trust score of the server, calculated by the trust provider, can be requested any time by a client. The client side trust profiling performed by the trust provider increase the resilience of the system against possible behaviour-based trust system attacks.

Trust can be expressed both quantitatively and qualitatively. Since this thesis is about calculating trust scores based on solid evidence in the form of past behavior and interaction, the concept of trust score is limited to measurable trust in this study. An ideal system model would have multiple trust providers performing trust score calculations using different algorithms, which adds variety and increases the robustness of trust score calculations. Multiple trust providers also create a competitive environment that forces them to provide high quality and accurate trust services in order to increase reliability. However, in this thesis, for ease of simulation, only a single trust provider is used which calculates the trust score using a set algorithm. Since the trust calculation is done by a single entity, the algorithm used to obtain accurate and reliable results becomes important.

To determine the trust score calculation algorithm to be used, implementations in a framework that supports various trust calculations provided by [13] are examined. It is claimed that one of the most common forms of scoring mechanism is averaging. [13] argues that averaging is very robust because the calculated trust score is reliable when the amount of feedback is high enough, and more importantly, it takes a significant amount of effort to generate significant score changes. They also state that averaging also has its challenges. One example would be the ignorance of temporal information tied to the order in which feedbacks are

recorded on the blockchain.

The trust calculation algorithm proposed in this thesis to assign the trust score of the central server is weighted averaging. Assuming a continuous satisfaction feedback function $f : X \rightarrow [0, 1]$ that links an interaction to its feedback where 1 is very satisfied and 0 is completely dissatisfied. Where f is the feedback rating the IoT device left for the server for a single interaction, e is the evidence of the interaction, w is the individual trust of the client as calculated by the trust provider, and n is the total number of feedbacks given for a server. The average weighted trust scoring algorithm μ used to calculate the trust score of the server is defined by:

$$\mu(\alpha) = \frac{\sum_i^n \phi(f, e) \cdot w}{n} \quad (1)$$

Similarly, averaging is used to calculate the trust of the clients based on the consistency of their feedback based on the evidence of the interaction. A continuous consistency function $\phi : F, E \rightarrow [0, 1]$ is used, which checks the consistency of the feedback f against its evidence e . If m denotes the total number of feedbacks given by a client, the average trust scoring algorithm η used to calculate the trust score of a client is defined by:

$$\eta(\beta) = \frac{\sum_i^m \phi(f, e)}{m} \quad (2)$$

An IoT client's individual trust score is privately logged by the trust provider but may also be made public in a decentralised system architecture. These individual trust scores are taken into account when calculating the trust score of the CoAP server. During averaging, each feedback an IoT device gives is weighted according to its trust. With the inclusion of client-side trust profiling, the resilience of the trust system against trust based-attacks in the network is increased.

CHAPTER IV

PERFORMANCE EVALUATION

This chapter will present information and discussion of the IoT network simulation performed to measure the effectiveness of the proposed security protocol and trust management model. The network simulation includes a CoAP server and 30 IoT nodes. Using the device information acquired on the connection phase, the server rejects certain devices and only allows devices which satisfy certain specifications to enter the network. The server also performs real-time autonomous anomaly detection using pre-trained ML algorithms to detect malicious or anomalous network traffic and blacklists the suspected IoT nodes.

Network simulation performed in this thesis is considered in three different aspects. The simulation of the proposed secure protocol, the autonomous anomaly detection and blacklisting, and the behavior of non-anomalous IoT nodes and the server in the network and their trust score assignments are considered.

This chapter will provide the dataset used to train the classifiers, compare different ML algorithms based on their performance and suitability for real-time anomaly detection, and present the performance of the proposed trust management model against various trust-based attacks. The simulation environment, the simulation results, as well as how the preferred classifier performed on the simulation are also presented.

4.1 Dataset Description

To simulate the network traffic inside an IoT network, the IoTID20 dataset is used [42]. The data is generated using a smart home environment which includes both benign and malicious devices. The dataset contains over 625,000 instances which consists of 80 network features and 3 label features which are shown in Table 2.

The creators of the IoTID20 dataset claims that the most important aspect of

Table 2: Label features of the IoTID20 dataset.

<i>Binary</i>	<i>Category</i>	<i>Sub – Category</i>
Normal Anomaly	Normal DoS (Denial of Service) Mirai MITM (Man in the Middle) Scan	Normal Syn Flooding Brute Force HTTP Flooding UDP Flooding ARP Spoofing Scan Host Port Scan OS

their dataset is that it is a great replication of modern day IoT device communications [42]. The dataset is also among the few publicly available IoT intrusion detection datasets. The features inside the data are ranked using the Shapira-Wilk algorithm to measure the regularity of the instance distribution based on their features. Based on this analysis, they claim that more than 70% of the features ranked above 0.5, stating that these high ranked features will improve the classification capability of ML classifiers and techniques.

The subset of data used for the network traffic simulation in this thesis is generated by randomly selecting 20,000 instances from each of the five main categories. In order to use the data in the simulation, additional features are added for ease of use; these features are not used by the classifiers during the training phase. The data is also revised to be compatible with the CoAP protocol.

4.2 Performance Evaluation of the Classifiers

Performances of the classifiers trained on the IoTID20 data may be observed in Tables 3 and 4, where the best performing classifiers on the data are RF, DT and Stacking algorithms. Even though the performance of the KNN classifier is similar with the RF and DT classifiers, [43] argues that the KNN algorithm is not suitable for critical real-time systems where high amounts of training instances are present. Also, the KNN algorithm requires significant computational resources as it requires calculation of all the distances between a new sample x and the entire training

space. Lastly, the GNB classifier was the worst performing classifier among all the algorithms considered.

Table 3: Macro Avg. Performances of the classifiers trained on category as target label.

<i>Classifier</i>	<i>Accuracy</i>	<i>Precision</i>	<i>Recall</i>	<i>F – 1Score</i>
Random Forest	0.90	0.93	0.90	0.90
Decision Tree	0.90	0.93	0.91	0.91
Stacking	0.91	0.93	0.92	0.92
K-Nearest Neighbor	0.87	0.91	0.87	0.86
Gaussian Naïve Bayes	0.56	0.66	0.56	0.56

Table 4: Macro Avg. Performances of the classifiers trained on sub-category as target label.

<i>Classifier</i>	<i>Accuracy</i>	<i>Precision</i>	<i>Recall</i>	<i>F – 1Score</i>
Random Forest	0.82	0.73	0.74	0.71
Decision Tree	0.85	0.78	0.77	0.74
Stacking	0.89	0.86	0.85	0.84
K-Nearest Neighbor	0.78	0.71	0.72	0.69
Gaussian Naïve Bayes	0.48	0.46	0.43	0.40

Among the best performing classifiers, namely the RF, DT and Stacking algorithms, the Stacking classifier is preferred in this thesis for the purpose of network simulation. However, it is important to mention that the Stacking classifier is an ensemble classifier and therefore it requires drastically more computational power and resources when compared with the other classifiers. Nonetheless, the difference in performance between the stacking algorithm and other classifiers cannot be neglected. As may be observed from Figure 4, the Stacking classifier appears to be by far the best classifier for detecting scan port OS attacks on the IoTID20 data, outperforming other classifiers significantly.

The classification time of the algorithms is also considered for real-time use. The Stacking classifier takes 0.06 s to classify a single instance, whereas the DT

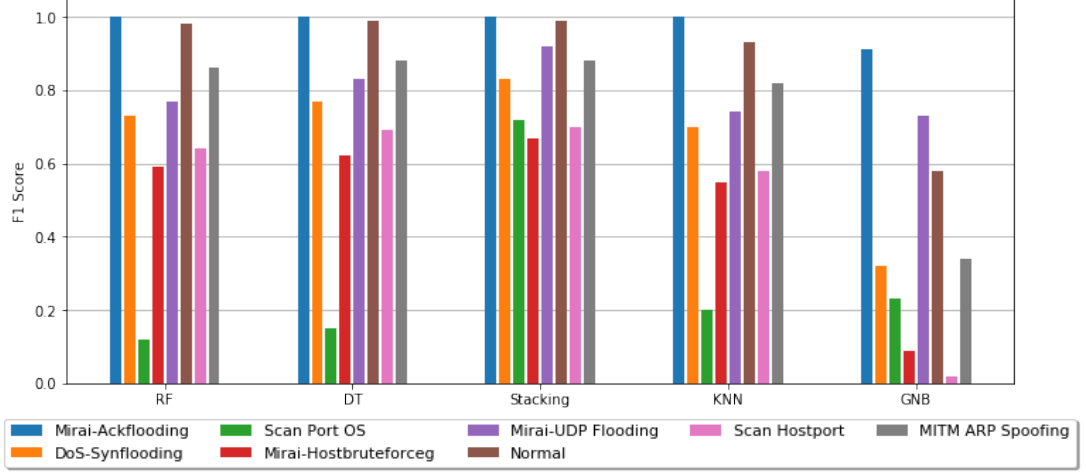


Figure 4: F1 Scores of the classifiers on the test data for sub-category label.

classifier takes only 0.01 s. This result is to be expected since the Stacking classifier is computationally more expensive. Nonetheless, the Stacking algorithm is preferred, as the difference between the two algorithms is quite negligible for the network simulation performed in this thesis. But, it is important to note that if the number of IoT devices were to increase significantly in the network, a CoAP server using the Stacking classifier may experience problems detecting anomalous network traffic in real-time. In such a case DT algorithm should be preferred.

4.3 Network Simulation

This section will provide the setup and results of the network simulation performed in detail for both the proposed security protocol and the trust management model. All network simulations are performed using Python 3.7.

4.4 Network Simulation of the Anomaly Detection Model

To simulate the network, a central CoAP server, another server to act as the CA, and 30 IoT devices as the client nodes were used. The IoT devices range from simple and lower complexity devices such as smart cameras, controllers and hubs, smoke alarms and printers, to more complex devices such as laptops, PCs,

phones, and tablets. The simulation assumes that none of these devices are previously known by the CoAP server. Therefore, each of these devices perform the registration phase to obtain a unique client ID and then use the connection phase to authenticate and establish a secure connection with the CoAP server.

A total of 30 IoT devices, which are used to simulate the IoT network, consists of 25 victim and 5 malicious devices. All malicious devices are high complexity devices. The number of devices used within the network simulation with their roles and computational complexities are shown in Table 5.

Table 5: Number of high-low complexity devices with their roles in the network simulation.

<i>Role</i>	<i>Complexity</i>	<i>Number of devices</i>
Victim	High	7
Victim	Low	18
Malicious	High	5

After going through both phases of the proposed security protocol and establishing a secure session with the CoAP server, each device starts generating a network traffic. Each request sent by the IoT devices were chosen from within the test data as discussed within the dataset section. Each device picks a random packet to send from within its assigned instances in the test data. For ease of implementation and simulation, the information within these packets are known by the CoAP server. This information is used by the server to perform real-time anomaly detection using the trained stacking classifier as discussed before.

The CoAP server adopts both a blacklisting and a whitelisting approach. Only high complexity devices such as PCs, phones, laptops, and tablets are whitelisted in simulation performed. Therefore, after the CoAP server receives the type and OS information of each device during the connection phase of the handshake protocol from I_C in the 7th message of the connection phase. The connection of devices which do not satisfy the whitelist policy described before are declined. Thus, the devices which satisfy the above conditions can generate a secure session with the

CoAP server and are registered to the IoT network. Currently the type of a device and its respective OS is enough to be added into the IoT network. However, more specific whitelist policies can easily be included for different scenarios, if needed.

The network traffic each individual IoT device generates within the simulation follows a Poisson probability distribution, which is widely used for network and communication traffic simulations [44]. The Poisson distribution expresses the probability of a given number of events occurring in a fixed duration of time. It assumes that these events occur independently with an average rate. Poisson distribution can be expressed using the equation below where x denotes the number of occurrences and λ denotes the average rate of occurrences. A Poisson distribution with $\lambda = 3$ packets sent per second by each client is used for the network simulation in this paper. It is important to mention that the λ can be any value as long as the CoAP server can handle the generated network traffic.

$$f(x) = \frac{\lambda^x}{x!} \cdot e^{-\lambda} \quad (3)$$

It is to be noted that the packet collision, loss, and faulty packets are omitted during the network simulation for convenience. Due to the whitelist policy, devices with lower complexity are denied connection by the server. Thus, only 12 devices can join the IoT network and begin generating the network traffic.

Table 6: Performance of the CoAP server on detecting anomalous packets correctly.

<i>Device Type</i>	<i>Number of Devices</i>	<i>Total Packets</i>	<i>Misclassified</i>	<i>Accuracy</i>
Victim	7	3871	66	98.29%
Malicious	5	2720	1	99.96%

The CoAP server feeds the information about each packet within the IoT network traffic into the classifier for autonomous anomaly detection in real time. The binary performance of the classifier, which can be seen from Table 6, is measured as the classifier’s ability to detect whether a packet is an anomaly or not. From

Table 6, it is shown that the classifier performs almost perfectly when detecting anomalous packets, with a false negative rate of only 0.038%. The classifier classified some packets in the victim network traffic as anomalous, which can be seen from the misclassified instances detected from victim devices, yielding a false positive rate of 1.71%. The overall accuracy of the classifier in the binary setting is 99.98%, misclassifying only 67 packets out of 6591 packets in the IoT network simulation. The CoAP server inside the proposed network simulation setup was able to blacklist all the malicious devices with perfect accuracy whereas none of the benign devices were blacklisted by accident.

Table 7: Performance of the CoAP server on correctly classifying packets inside the network.

<i>Device Type</i>	<i>Number of Devices</i>	<i>Total Packets</i>	<i>Misclassified</i>	<i>Accuracy</i>
Victim	7	3871	66	98.29%
Malicious	5	2720	254	90.06%

The classifiers performance on correctly classifying each packet can be seen from Table 7. Comparing both Table 6 and Table 7, since there is no additional classification of normal packets, the number of misclassified instances for victim devices remain the same. Nonetheless, there are significant amounts of misclassified packets generated by the malicious devices. However, this does not mean that the classifier fails to detect anomalous packets, but rather fails to classify the type of the anomalies.

Considering both Table 6 and Table 7, the ability to detect the anomalous packets in the network traffic is enough for the CoAP server to maintain the security of the IoT network, and the classification of anomalies is not a high priority. Therefore, it can be said that the Stacking classifier performs quite remarkably with an overall accuracy of 99.98%. Since the accuracy of the classifier is very high, a blacklisting approach can be used by the CoAP server to prevent malicious devices from generating network traffic within the IoT network.

Current limitations of the proposed system model include the assumption of

secure verification between the IoT devices and the CA for server authentication. The model assumes secure communications between the two which may be achieved through other handshake protocols like TLS or a long term pre-shared secret which is inserted into the IoT devices and the CA. Both assumptions have their downsides in the proposed model. TLS uses traditional session key generation and public key encryption, which requires high computational power and hence not suitable for all IoT devices. Similarly, using a long term pre-shared secret is not suitable for the proposed use case and system model. Since any device capable of completing the proposed security protocol may enter the IoT network, distributing the pre-shared secret to the new devices poses a challenge.

The deficiencies of the proposed network simulation setup include the behavior of the malicious IoT nodes. The attacker IoT nodes within the network simulation generate high amounts of malicious network traffic which makes it relatively easy for the server to detect the malicious nodes since the classifier used for autonomous anomaly detection has a high performance as presented before. This may not be the case in a real-life scenario where the malicious devices actively try to impersonate benign devices by generating small amounts of malicious network traffic to avoid detection.

4.5 Network simulation of the Trust Model

A setup similar to the one used in the network simulation of the proposed security protocol is used to simulate the trust model. The setup includes a single CoAP server, one server each to implement the CA, trust provider, and the blockchain, and 30 IoT devices. All IoT devices are assumed to be of high complexity to join the network due to the whitelisting policy discussed before. Similar to the security protocol, the simulation does not assume previously known devices, so all the devices perform both phases of the security protocol to establish a secure session. All devices within the network do not perform protocol-based attacks to avoid getting blacklisted by the CoAP server. Instead, malicious devices inside the

network perform trust-based attacks, as discussed in the adversary model of our trust model. Different cases which include malicious devices of different behaviors are considered and multiple simulations are performed to assess the performance of the trust provider and resilience of the proposed system against different types of trust-based attacks. The network simulations performed assume a benign server and therefore, all trust-based attacks attempt to bad-mouth the server, although the reverse can also be simulated with similar results.

All the IoT devices in the network request the current trust score of the CoAP server from the trust provider after each interaction. A trust threshold of 0.5 is used by all clients, where if the trust score of the server provided by the trust provider is less than 0.5, the device leaves the network. The trust provider updates the trust of the CoAP server every 5 seconds. The trust provider also provides clients when the current trust score was last updated. In all network simulations performed, the ratio of benign and malicious clients is kept at 1:1 (half of the nodes in the network are benign and the other half are malicious nodes). The network traffic each IoT device generates within the simulation again follows a Poisson distribution, where 30 devices generate approximately 300 requests each.

4.5.1 Explicit Attack Scenario

In this simulation, the malicious nodes bad-mouth on every interaction and do not care about their trust scores. The distribution of the attacker nodes and benign nodes is 1:1. The performance of the trust provider in these settings is shown in Figure 5.

Here, the black curve represents the average trust score of the CoAP server based on just feedback available on blockchain, whereas the red curve represents the trust score of the CoAP server calculated by the trust provider. The actual average trust score of the CoAP server based on feedback available on blockchain is 0.45, whereas the trust score calculated by the trust provider at the end of this simulation is recorded as 0.81. Based on these results, it can be concluded that the proposed trust model is resilient against nodes that perform attacks with no

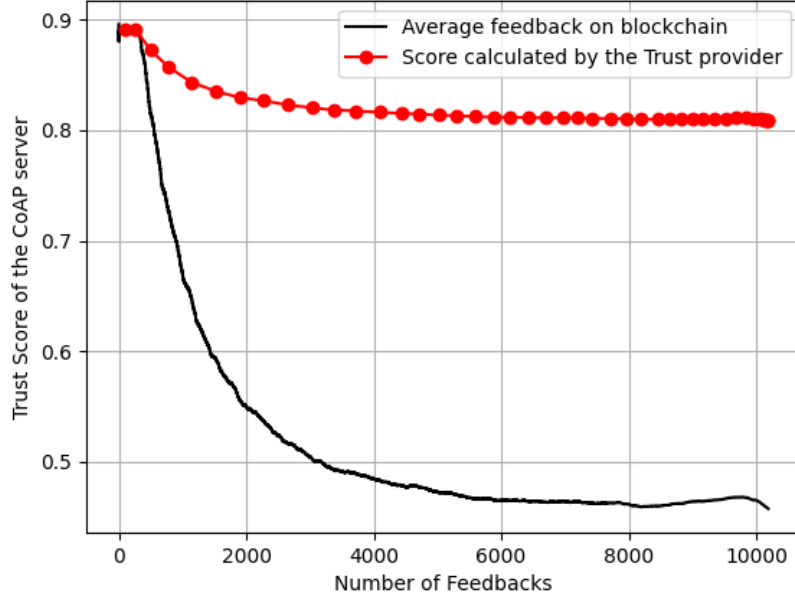


Figure 5: Performance of the trust provider against explicit attacks.

consideration for their trust scores and such nodes can easily be detected by the trust provider.

4.5.2 Random Attack Scenario

In this simulation, malicious nodes randomly give inconsistent feedback, again without considering their trust score. The distribution of the attacker nodes and benign nodes is 1:1. The performance of the trust provider in these settings is shown in Figure 6.

The actual average trust score of the CoAP server based on just feedback available on blockchain is 0.65, whereas the trust score calculated by the trust provider at the end of this simulation is recorded as 0.73. It is seen that the average feedback score calculated from the blockchain is higher than the previous attack scenario involving explicit attacks. This is to be expected because random attacks may sometimes yield consistent feedback. From the results shown in Figure 6, the trust score calculation of the trust provider still yields better results. Therefore, one may conclude that the proposed trust model is resilient against nodes that

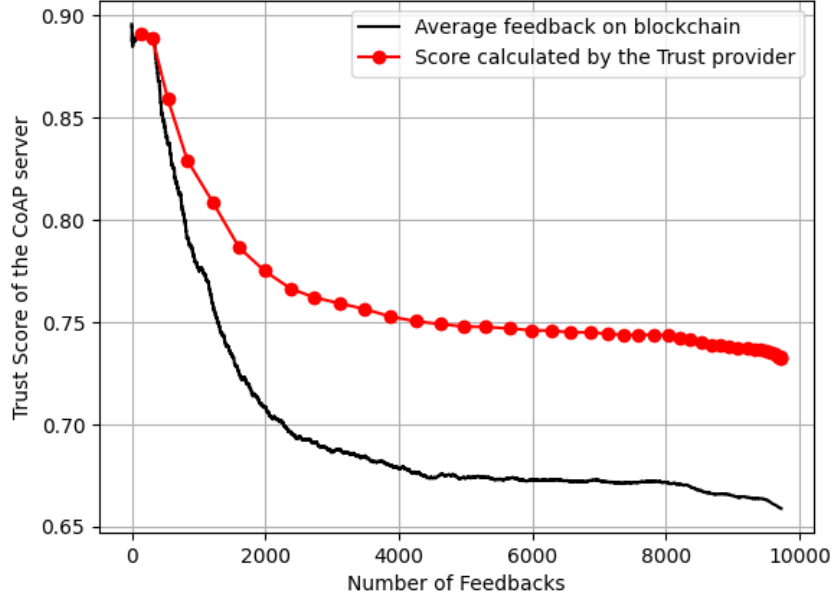


Figure 6: Performance of the trust provider against random attacks.

perform random attacks.

4.5.3 Opportunistic Attack Scenario

In this simulation, malicious nodes try to attain a high trust score by behaving like benign nodes and giving consistent feedback. After obtaining a high trust score, they then start acting maliciously. The attacker nodes score the first 33.3% of their interactions like benign nodes, and then perform bad-mouthing on the other 66.6% of their interactions. The distribution of the attacker nodes and benign nodes is 1:1. The performance of the trust provider in these settings is shown in Figure 7.

The actual average trust score of the CoAP server based on just feedback available on blockchain is 0.55, whereas the trust score calculated by the trust provider at the end of this simulation is recorded as 0.68. Even against opportunistic attacks, the trust provider still outputs a higher trust score for the benign CoAP server as seen in Figure 7. One would expect that the opportunistic attacks would be more effective on the trust system as they are harder to detect. But, when

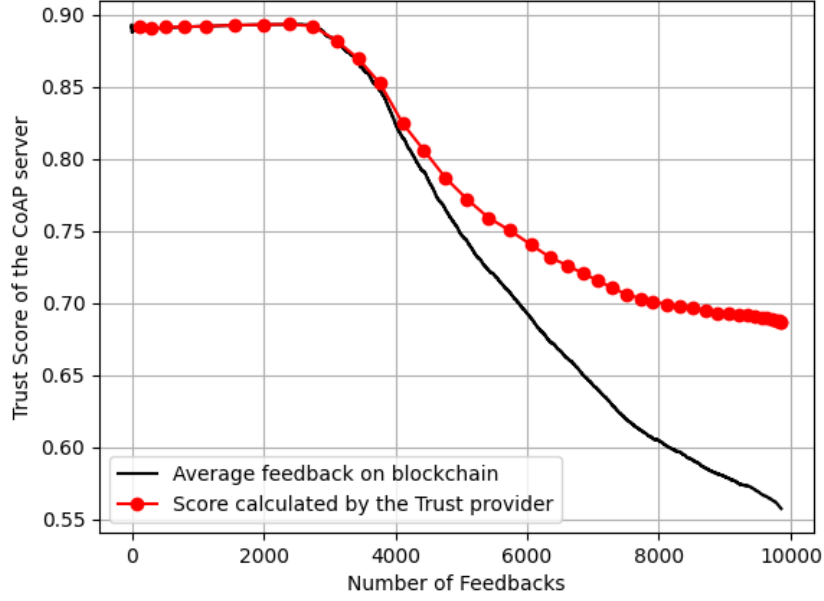


Figure 7: Performance of the trust provider against opportunistic attacks.

compared with the random attacks, simulation results show that random attacks are harder for the trust provider to sustain.

4.5.4 Collusion Attack Scenario

In this simulation, malicious nodes act together. They start out as benign nodes, but suddenly start acting malicious together and suddenly stop acting malicious together. The nodes which perform collusion attacks in this simulation start bad-mouthing 60 seconds after the simulation starts and stop bad-mouthing 120 seconds into the simulation (the simulation itself takes approximately 220 seconds). The distribution of the attacker nodes and benign nodes is 1:1. The performance of the trust provider in these settings is shown in Figure 8.

The actual average trust score of the CoAP server based on just feedback available on blockchain is 0.71, whereas the trust score calculated by the trust provider at the end of this simulation is recorded as 0.75. The trust scores are quite close to each other because after the collusion attack ends, the attacker nodes starts acting like benign nodes again, which in turn increases the trust score of the

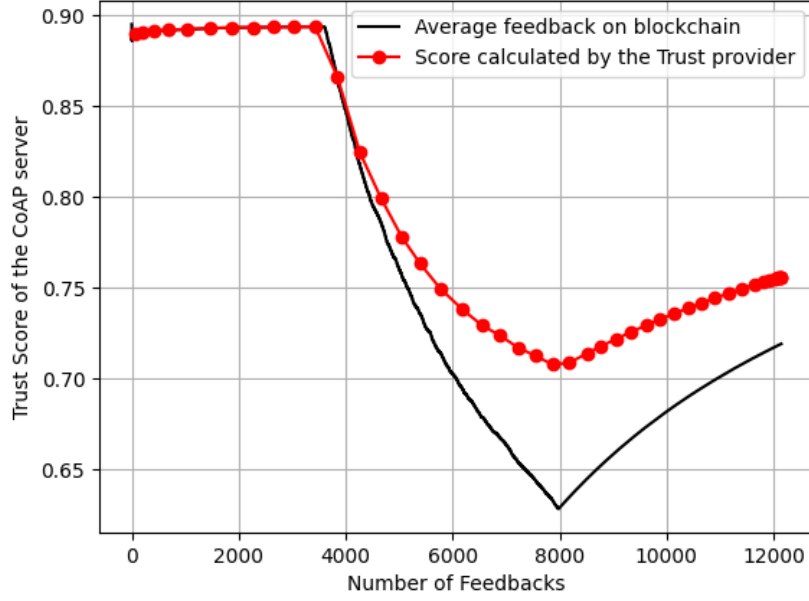


Figure 8: Performance of the trust provider against collusion attacks.

CoAP server through the end of the simulation. Therefore, in this case, the trust score of the server at the end of the collusion attack needs to be considered instead. From Figure 8, it can be seen that the highest discrepancy between the trust score calculated by the trust provider and the average of the feedback happens when the total number of feedback given reaches approximately 8000, which is when the collusion attack concludes. The average trust score of the server at this time is 0.62, whereas the trust score calculated by the trust provider is 0.71. Based on these results, it can be seen that the trust score calculated by the trust provider is resilient against collusion attacks as well.

4.5.5 Discussion of the Trust Model Simulation

Considering the results of the trust model simulations, it can be said that the trust score profiling performed on the client nodes by the trust provider provides resilience against various trust system attacks. The simulation results show that the trust score of the benign CoAP server never falls below the threshold of 0.5, even if half of the nodes in the network are malicious. It is important to note

that capturing half of the nodes in a real-life network is quite difficult. Because the addition of a client-side trust profile demonstrates increased resilience to trust system attacks, this approach can be used in centralized networks to increase the consistency of trust providers in a system architecture similar to the one proposed in this thesis.

Current limitations of the proposed system model include the assumption that evidence of an interaction cannot be manipulated. Of course, once evidence is recorded on the blockchain, it is almost impossible to tamper with, but evidence can be erroneously recorded directly on the blockchain by the nodes giving feedback. This can be prevented with the usage of smart contracts as proposed in [13], where the evidence linking an interaction to its feedback is not provided by the nodes but instead recorded onto the blockchain if certain conditions are met within that interaction by utilizing smart contracts. However, this approach would increase the complexity of the architecture and the smart contract system would require access to every variable of the service provided within an interaction between the nodes and the server.

Another limitation of the proposed trust system model involves the use of PoW for blockchain implementation. Using PoW can create challenges for nodes with low computational complexity and limit their ability to give feedback on each of their interactions. However, while not suitable for all IoT devices, there are benefits in using PoW in the recommended trust system model. Currently, there is no cost for nodes within the network to send requests and interact with the server. This means that there is nothing stopping attacking nodes from sending an unusual number of requests and performing good-bad mouthing on every interaction. Using PoW can limit the attacker nodes in such scenarios since they would require to expand considerable amounts of computational power.

CHAPTER V

CONCLUSIONS

This thesis presents a three-layer security framework for centralized IoT networks to provide security against protocol-, network-, and behavior-based attacks. A security protocol is proposed to provide communication privacy and mutual authentication. An autonomous anomaly detection using ML algorithms is proposed to provide security against network-based attacks from inside the network. And finally, a trust management model is proposed to increase resilience against behavioral attacks.

The protocol makes use of a PUF for session key generation, as well as whitelisting and blacklisting methods for ensuring the network security. To the best of our knowledge, the presented PUF-based handshake protocol is the first to perform blacklisting and whitelisting. Anomaly detection using ML techniques is utilized for blacklisting. An in-depth security analysis of the proposed handshake protocol is provided, where the resilience of the protocol against forgery, replay and MITM attacks is shown. The thesis also provides and discusses network traffic simulation carried out using the proposed handshake protocol. By gathering the specifications of IoT nodes, the CoAP server only allows nodes to join or remain in the IoT network that satisfy certain specifications using whitelisting and blacklisting methods. Additionally, a comparison of various ML algorithms performances and their fitness for real-time anomaly detection is provided. Amongst the five ML algorithms considered, the Stacking algorithm is shown to display the highest level of accuracy of 99.98% in detecting anomalous data in the network. Based on the results of the network simulation performed, the CoAP server inside the proposed setup was able to blacklist all the malicious devices amongst the IoT clients with perfect accuracy.

The proposed trust management model, utilizing blockchain, is capable of consistent trust score assignment in a central network where trust of both clients and servers is ambiguous. Using client-side trust profiling and average weighted trust scoring algorithm, the resilience of the proposed trust system against various forms of trust attacks is shown with the network simulations performed.

Future work includes implementing a more comprehensive trust system suitable for distributed systems with multiple servers and trust providers, where trust providers may also be malicious.



REFERENCES

- [1] A. Rayes and S. Salam, “Internet of things from hype to reality,” pp. 1–35, 2019.
- [2] G. Santucci, “The internet of things: Between the revolution of the internet and the metamorphosis of objects,” *Vision and Challenges for Realising the Internet of Things*, vol. 1, pp. 11–24, 2010.
- [3] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, “Diot: A federated self-learning anomaly detection system for iot,” in *IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pp. 756–767, IEEE, 2019.
- [4] G. Fortino, F. Messina, D. Rosaci, and G. M. Sarnè, “Resiot: An iot social framework resilient to malicious activities,” *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 5, pp. 1263–1278, 2020.
- [5] C. A. Baykara, I. Şafak, and K. Kalkan, “Shapeiot: Secure handshake protocol for autonomous iot device discovery and blacklisting using physical unclonable functions and machine learning,” *3rd International Conference on Machine Learning & Applications*, vol. 11, no. 15, pp. 119–137, 2021.
- [6] D. F. Santos, H. O. Almeida, and A. Perkusich, “A personal connected health system for the internet of things based on the constrained application protocol,” *Computers & Electrical Engineering*, vol. 44, pp. 122–136, 2015.
- [7] T. A. Alghamdi, A. Lasebae, and M. Aiash, “Security analysis of the constrained application protocol in the internet of things,” in *Second International Conference on Future Generation Communication Technologies (FGCT)*, pp. 163–168, IEEE, 2013.
- [8] D. Rathod and S. Patil, “Security analysis of constrained application protocol (coap): Iot protocol,” *International Journal of Advanced Studies in Computers, Science and Engineering*, vol. 6, no. 8, p. 37, 2017.
- [9] Z. Shelby, K. Hartke, and C. Bormann, “The constrained application protocol (coap).” https://iottestware.readthedocs.io/en/master/coap_rfc.html, (Accessed May 2022).
- [10] A. Shamsoshoara, A. Korenda, F. Afghah, and S. Zeadally, “A survey on physical unclonable function (puf)-based security solutions for internet of things,” *Computer Networks*, vol. 183, p. 107593, 2020.
- [11] G. Chen, F. Zeng, J. Zhang, T. Lu, J. Shen, and W. Shu, “An adaptive trust model based on recommendation filtering algorithm for the internet of things systems,” *Computer Networks*, vol. 190, p. 107952, 2021.

- [12] S. Pal, M. Hitchens, and V. Varadharajan, "Towards the design of a trust management framework for the internet of things," in *13th International Conference on Sensing Technology (ICST)*, pp. 1–7, IEEE, 2019.
- [13] S. Pal, A. Hill, T. Rabehaja, and M. Hitchens, "A blockchain-based trust management framework with verifiable interactions," *Computer Networks*, vol. 200, p. 108506, 2021.
- [14] Y. L. Sun, Z. Han, W. Yu, and K. R. Liu, "Attacks on trust evaluation in distributed networks," in *40th Annual Conference on Information Sciences and Systems*, pp. 1461–1466, IEEE, 2006.
- [15] M. U. Hassan, M. H. Rehmani, and J. Chen, "Privacy preservation in blockchain based iot systems: Integration issues, prospects, challenges, and future research directions," *Future Generation Computer Systems*, vol. 97, pp. 512–529, 2019.
- [16] S. Nakamoto, *Bitcoin: A peer-to-peer electronic cash system*. Association of American Publishers (AAP), 2008.
- [17] E. Rescorla, "The transport layer security (tls) protocol version 1.3." <https://www.rfc-editor.org/rfc/rfc8446>, (Accessed May 2022).
- [18] Y. Yilmaz, S. R. Gunn, and B. Halak, "Lightweight puf-based authentication protocol for iot devices," in *IEEE 3rd international verification and security workshop (IVSW)*, pp. 38–43, IEEE, 2018.
- [19] A. Bhattacharyya, T. Bose, S. Bandyopadhyay, A. Ukil, and A. Pal, "Less: Lightweight establishment of secure session: A cross-layer approach using coap and dtls-psk channel encryption," in *IEEE 29th International Conference on Advanced Information Networking and Applications Workshops*, pp. 682–687, IEEE, 2015.
- [20] A. B. Rabiah, K. Ramakrishnan, E. Liri, and K. Kar, "A lightweight authentication and key exchange protocol for iot," in *Workshop on Decentralized IoT Security and Standards*, vol. 2018, pp. 1–6, 2018.
- [21] J. W. Byun, "End-to-end authenticated key exchange based on different physical unclonable functions," *IEEE Access*, vol. 7, pp. 102951–102965, 2019.
- [22] J. W. Byun, "A generic multifactor authenticated key exchange with physical unclonable function," *Security and Communication Networks*, vol. 2019, 2019.
- [23] M. N. Aman, K. C. Chua, and B. Sikdar, "Mutual authentication in iot systems using physical unclonable functions," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1327–1340, 2017.
- [24] A. Braeken, "Puf based authentication protocol for iot," *Symmetry*, vol. 10, no. 8, p. 352, 2018.
- [25] U. Chatterjee *et al.*, "Puf+ ibe: blending physically unclonable functions with identity based encryption for authentication and key exchange in iots," *Cryptology ePrint Archive*, 2017.

- [26] N. Li, D. Liu, and S. Nepal, "Lightweight mutual authentication for iot and its applications," *IEEE Transactions on Sustainable Computing*, vol. 2, no. 4, pp. 359–370, 2017.
- [27] Y. Ma, L. Yan, X. Huang, M. Ma, and D. Li, "Dtlshps: Sdn-based dtls handshake protocol simplification for iot," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3349–3362, 2020.
- [28] J. Park and N. Kang, "Lightweight secure communication for coap-enabled internet of things using delegated dtls handshake," in *International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 28–33, IEEE, 2014.
- [29] B. Anderson and D. McGrew, "Accurate tls fingerprinting using destination context and knowledge bases," *arXiv preprint arXiv:2009.01939*, 2020.
- [30] S. A. Ghasempouri and B. T. Ladani, "Modeling trust and reputation systems in hostile environments," *Future Generation Computer Systems*, vol. 99, pp. 571–592, 2019.
- [31] S. K. Prajapati, S. Changder, and A. Sarkar, "Trust management model for cloud computing environment," *arXiv preprint arXiv:1304.5313*, 2013.
- [32] S. E. A. Rafey, A. Abdel-Hamid, and M. Abou El-Nasr, "Cbstm-iot: Context-based social trust model for the internet of things," in *International Conference on Selected Topics in Mobile & Wireless Networking (MoWNeT)*, pp. 1–8, IEEE, 2016.
- [33] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198–208, 1983.
- [34] G. Kambourakis, C. Kolias, and A. Stavrou, "The mirai botnet and the iot zombie armies," in *MILCOM IEEE Military Communications Conference (MILCOM)*, pp. 267–272, IEEE, 2017.
- [35] S. Byers, A. D. Rubin, and D. Kormann, "Defending against an internet-based attack on the physical world," *ACM Transactions on Internet Technology (TOIT)*, vol. 4, no. 3, pp. 239–254, 2004.
- [36] J. M. Estevez-Tapiador, P. García-Teodoro, and J. E. Díaz-Verdejo, "Detection of web-based attacks through markovian protocol parsing," in *10th IEEE Symposium on Computers and Communications (ISCC'05)*, pp. 457–462, IEEE, 2005.
- [37] R. Xu, W.-l. Ma, and W.-l. Zheng, "Defending against udp flooding by negative selection algorithm based on eigenvalue sets," in *Fifth International Conference on Information Assurance and Security*, vol. 2, pp. 342–345, IEEE, 2009.
- [38] A. Sridharan, T. Ye, and S. Bhattacharyya, "Connectionless port scan detection on the backbone," in *IEEE International Performance Computing and Communications Conference*, pp. 10–pp, IEEE, 2006.

- [39] T. Mehmood and H. B. M. Rais, “Machine learning algorithms in context of intrusion detection,” in *3rd International Conference on Computer and Information Sciences (ICCOINS)*, pp. 369–373, IEEE, 2016.
- [40] P. Illavarason and B. K. Sundaram, “A study of intrusion detection system using machine learning classification algorithm based on different feature selection approach,” in *Third International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*, pp. 295–299, IEEE, 2019.
- [41] A. Srivastava, A. Agarwal, and G. Kaur, “Novel machine learning technique for intrusion detection in recent network-based attacks,” in *4th International Conference on Information Systems and Computer Networks (ISCON)*, pp. 524–528, IEEE, 2019.
- [42] I. Ullah and Q. H. Mahmoud, “A scheme for generating a dataset for anomalous activity detection in iot networks,” in *Canadian Conference on Artificial Intelligence*, pp. 508–520, Springer, 2020.
- [43] A. Starzacher and B. Rinner, “Evaluating knn, lda and qda classification for embedded online feature fusion,” in *International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, pp. 85–90, IEEE, 2008.
- [44] B. Chandrasekaran, “Survey of network traffic models,” *Washington University in St. Louis CSE*, vol. 567, 2009.