Towards Improved Trust in Threat Intelligence Sharing using Blockchain and Trusted Computing

Yichang Wu

Supervised by Dr. Brian Lee, Dr. Yuansong Qiao

A thesis presented for the degree of

Master of Science

Software Research Institute

Athlone Institute of Technology

June 2020

Author	**Degree**	**Time**
Yichang Wu	Master of Science	June 2020

Thesis Title

Towards Improved Trust in Threat Intelligence Sharing using Blockchain and Trusted Computing

Supervisor

Brian Lee, Yuansong Qiao

Abstract

Threat Intelligence Sharing is one of the most promising approaches to resist ever-growing cyber attacks. Pressing challenges including concerns about trust, privacy, negative publicity, policy/legal issues and expense of sharing inhibit effective sharing. Among all these sharing requirements, open trust issues are most crucial and fundamental prerequisites. Blockchain Technology has been widely adapted in data sharing market such as IoT Data sharing, Health Information sharing and so on. Trusted Computing provides an isolated secure trustworthy execution environment.
This thesis **AIMs** to investigate the use of Blockchain Technology and Trusted Computing as a mechanism to address open trust issues of threat intelligence sharing.

Objectives:

1. To understand the drivers, benefits and challenges for organizations to exchange threat intelligence.

2. To investigate and enumerate the functions of Blockchain that facilitate exchange of data by studying existing examples in other domains.

3. To identify how Blockchain and Trusted Computing can motivate and enable threat intelligence sharing by emphasizing its benefits and overcoming its challenges.

4. To design a trust enhanced threat intelligence sharing system using Blockchain and Trusted Computing.

5. To prototype and evaluate the feasibility of proposed solution.

This work surveys the benefits and challenges of threat intelligence sharing, studies several Blockchain based systems in the data marketplace, and identifies three types of trust requirements for threat intelligence sharing. Afterwards, I describe a novel trust enhancement framework -TITAN- for decentralized sharing based on the use of P2P reputation systems to address open trust issues. Our design uses blockchain and Trusted Execution Environment technologies to ensure security, integrity and privacy in the operation of the Threat Intelligence Sharing reputation system.

Keywords

Cyber Security, Threat Intelligence Sharing, Reputation, Blockchain, TEE

Declaration

I hereby certify this material, being submitted for assessment on the programme of study leading to the award of Master of Science is entirely my own work and has not been taken from the work of others, and to the extent that such work has been cited within the text of the work.

Student ID: A00257949

Name: Yichang Wu

Signature:

Date:

Acknowledgments

First of all, this project is funded by Government of Ireland - International Education Scholarship (GOI-IES) 2018, 2019. I would like to extend my sincere gratitude to my sponsor - Government of Ireland. I want to say thanks to government officials and AIT staffs who manage the funding.

Then, I owe my deepest gratitude to my main supervisor, Dr. Brian Lee. His professional expertise and extensive knowledge guide me and the project passing through most critical points. His careful and comprehensive review helps a lot to this composition. I am also extremely grateful to my second supervisor Dr. Yuansong Qiao. The conversation with Dr. Qiao has always inspired many creative ideas and insights.

Lastly, I will offer my special thanks to my family. They raise me up, so I can stand on mountains.

Publication

The following list outlines the dissemination of this work to date:

1. Publication - Yichang Wu, Yuansong Qiao, Yuhang Ye and Brian Lee, “Towards Improved Trust in Threat Intelligence Sharing using Blockchain and Trusted Computing,” in *The International Symposium on Blockchain Computing and Applications*, BCCA, 2019.

Contents

List of Figures

List of Tables

List of Abbreviations

BTC Bitcoin. 16

CA Certification Authority. 25

DoD Department of Defense. 19

DSR Design Science Research. 5

EE Execution Environment. 22

EOA Externally Owned Account. 17

H2020 Horizon 2020. 63

IDEA Intrusion Detection Extensible Alert. 50

IEP Information Exchange Policy. 33

IoT Internet of Things. 26

IT Information Technology. 3

KE Knowledge Exchange. 2, 40

OMTP Open Mobile Terminal Platform. 22

P2P Peer-to-Peer. 10

PII Personally Identifiable Information. 32

QoI Quality of Indicators. 37

SCONE Secure Container Environment. 57

SED Self-Encrypting Drives. 19

SGX Software Guard Extensions. 23

STIX Structured Threat Information eXpression. 32

TCB Trusted Computing Base. 19

TCG Trusted Computing Group. 19

TCPA Trust Computing Platform Alliance. 19

TCSEC Trusted Computer System Evaluation Criteria. 19

TI Threat Intelligence. 1

TIP Threat Intelligence Platform. 2

TIQ Threat Intelligence Quality. 2

TITAN Threat Intelligence sharing Trust enhANcement. 45

TLP Traffic Light Protocol. 35

TLS Transport Layer Security. 58

TNC Trusted Network Communications. 19

TPM Trusted Platform Module. 19

TPS Transactions Per Second. 29

TTP Tactics, Techniques and Procedures. 30

URL Uniform Resource Locator. 27

UTXO Unspent Transaction Output. 17

Chapter 1

Introduction

1.1 Background

The incidence of cyber attacks and threats is continually growing, as can be attested by the numerous threat and malware reports on the Internet such as those from the leading security solution vendors. Organizations are being encouraged to share *Threat Intelligence (TI)* i.e. information of received attacks and other cyber security experiences with each other, as a means to counter such threats. Such sharing will give organizations a broader view of the current cybersecurity picture, e.g. at a regional, national or sectoral level, and thus increase their level of cyber situational awareness. This will in turn improve their attack preparedness (security posture) and thus decrease their risk of cyber attack.

Despite the clear benefits to be gained, and the presence of the large number of sharing platforms, there are still **substantial barriers remaining to sharing threat intelligence**. These include organizations concerns about privacy (confidentiality of information), negative publicity, policy and legal issues and cost amongst others [1]. Furthermore, there are fears that not all participants will contribute equally i.e. some parties may 'freeride' and consume more than they contribute [2].

Threat intelligence is normally shared between members of well-defined groups or *communities* with varying degrees of openness in sharing. Communities may be *private* i.e. closed with sharing only between the members of the community or *public* i.e. fully open with all information available to anyone; or they may be somewhere in between [3]. Additionally, some companies provide information commercially. Recently there is evidence of a move away from the existing semi-static community model towards a more decentralized, dynamic based model - inspired (and enabled) in large part by the increasing trend, in many fields, to share data and information via blockchain based markets. One such example, based on Hyperledger, is TRADE from IBM [4]. Other examples can be found in the number of startups - such as Polyswarm [5] - establishing markets for threat intelligence sharing systems. I fully expect this trend to continue and, furthermore, to see current systems evolve in this direction also. A possible blueprint for how a future hybrid centralized/decentralized threat intelligence sharing environment might play out is the *Knowledge Exchange* (KE) concept from Serrano [3].

1.2 Motivation

Although sharing of threat intelligence is happening its effectiveness is greatly held back due to trust related issues [1, 3, 2, 6, 7]. Tounsi and Rais identify untrusted participants as one of the crucial obstacles for sharing though their survey [1]. The motivation of sharing will be discouraged when only few parties are active without getting much in return from the other parties. Sauerwein et al. emphasize that organizations must trust the sharing platform [2]. They suggest the access to these platforms must be restricted in order to avoid any unauthorized access. Johnson believe the trustworthiness of the recipient also should be taken into consideration, otherwise the information may be misused [6].

Trust is a broad topic with sometimes subtle shades of meaning. I identify three type of trust concerns when considering TI sharing: *Trust between Participants, Trust of TI Quality (TIQ) and Trust in the TI Platform (TIP).* All of these will be of concern in the emerging decentralized TI sharing environment.

Trust between participants is needed for participants to engage in sharing in the first place. Trust in the platform helps provide reassurance about engagement in the community through the provision of Information Technology (IT) security mechanisms to support confidentiality, integrity and privacy. Trust in TI quality helps information receivers more efficiently deal with large volumes of TI by selecting only data with required quality levels.

None of the emerging solutions that I have seen fully addresses all of these issues. TRADE does discuss a reputation management system to help address participant trust but few details are provided and they don't at all mention data quality trust [4]. Nor is this issue addressed satisfactorily by the various startups in the area, many of whom also neglect mechanisms for participant trust management. Al-Ibrahim [8] describes an architectural approach for assessing TI quality but gives no implementation details and does not at all consider blockchain based systems.

1.3 Research Objectives and Questions

To address these trust issues, Blockchain Technology is deemed as an approach to remove the central trust authority, and Trusted Computing ensure the computing process tamper-proof. This thesis investigates the incorporation of Blockchain Technology and Trusted Computing as a means to achieve the trust requirements of Threat Intelligence Sharing.

1.3.1 Research Objectives

There are five research objectives this thesis tries to achieve:

1. To understand the drivers, benefits and challenges for organizations to exchange threat intelligence.

2. To investigate and enumerate the functions of Blockchain that facilitate exchange of data by studying existing examples in other domains.

3. To identify how Blockchain and Trusted Computing can motivate and enable threat intelligence sharing by emphasizing its benefits and overcoming its challenges.

4. To design a trust enhanced threat intelligence sharing system using Blockchain and Trusted Computing.

5. To prototype and evaluate the feasibility of proposed solution.

1.3.2 Research Questions

1. What advantages does the use of blockchain bring to threat intelligence sharing?

2. How could trusted computing help addressing the trust requirements of threat intelligence sharing?

3. Is is feasible to create a practical trusted treat intelligence sharing system using Blockchain Technology and Trusted computing?

Chapter 2

Methodology

The methodologies applied in computer science are varied and don't always match to those practised in the natural and social sciences and humanities.

Whittemore and Melkus give an overview of research process which consists of five phases: Conceptual phase, Design and Planning phase, Empirical phase, Analytic phase, Dissemination phase [9]. In the Design and Planning phase, they generalize two types of research design: Qualitative design - aims to describe or understand a phenomenon. Quantitative design - aims to measure the relationships among variables or examine the effects of an intervention.

In Simon's perspective, "Natural Science" is different from "Design Science" [10]. The former is knowledge about objects or phenomenon in the natural and societal world to describe and explain how they behave and interact with each other. The latter is knowledge about the ***design*** of artifacts meeting certain desired goals. "Design Science Research" fails to fall into above two categories.

Vaishnavi depicts a Design Science Research (DSR) Process Model in Figure 2.1 [11]:

The DSR process model comprises five steps.

1. Awareness of Problem: To survey and analyse the problem domain and the current solution domain.

2. Suggestion: To give a tentative design based on previous outcome.

3. Development: To develop and implement the above design. The feedback

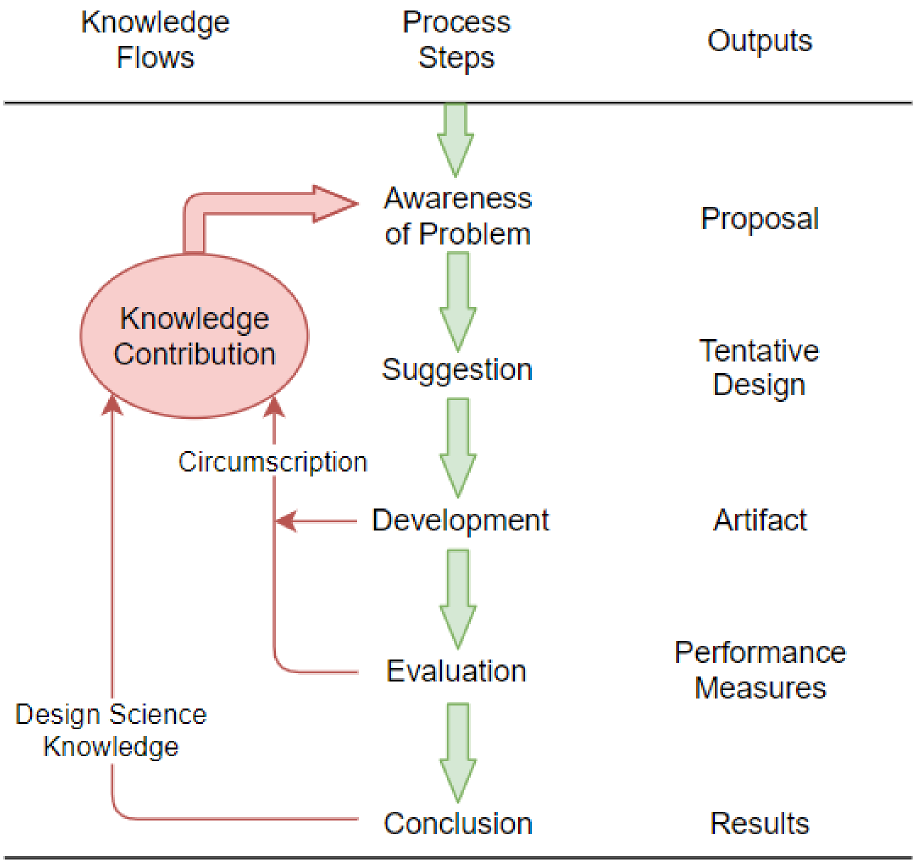


Figure 2.1: Design Science Research (DSR) Process Model

of development will contribute knowledge and increase problem awareness.

4. Evaluation: Once constructed, the artifact will be evaluated. Any deviations from expectations may lead to another round of design, development and evaluation.

5. Conclusion: To conclude the whole research progress and disseminate the knowledge generated.

This is the methodology generally used in computer science. It gives birth to various of innovative protocols, algorithms and artifacts.

I apply the methodology as follows: I will analyse the motives and barriers of Threat Intelligence Sharing (Objective 1) and study the current Blockchain and Trusted Computing implementations (Objective 2). Then I will design a specific solution to embody the desired goals (Objective 3 and 4). Afterwards, I validate it by implement and evaluate such a solution (Objective 5).

Chapter 3

Literature Review

3.1 Introduction

In this chapter, I review the underlying technologies that I use as a base. I also review literature relevant to related works. This includes the topics that form part of Objective 1, 2 and 3, i.e., I consider:

- Benefits and Challenges for Threat Intelligence Sharing (Objective 1). I introduce the background, benefits and challenges of Threat Intelligence Sharing in section 3.4.

- Blockchain Technology basics (Objective 2). In section 3.2.2, I state the main concepts of Blockchain Technology which is the kernel of our project. And I draw four main characteristics of Blockchain.

- Trusted Computing basics (Objective 2). Section 3.2.3 presents the background and concepts of Trusted Computing.

- Notion of Trust (Objective 3). To identify the trust requirements, I will clarify the notion of trust in section 3.2.1.

- Data sharing using Blockchain (Objective 2 and 3). With the basic understanding of Blockchain, I investigate several Blockchain implementations in Data Marketplace to learn the advantages of Blockchain in data sharing. It is depicted in section 3.3.

- Blockchain in Threat intelligence Sharing (Objective 3). I discuss how Blockchain will work to mitigate the challenges of Threat Intelligence Sharing, as shown in section 3.5

3.2 Trust Enablers

3.2.1 Notions of Trust

Trust is, in general, a complex and multi-faceted notion. According to Josang [12] it "is a directional relationship between two parties that can be called trustor and trustee" where the former is considered a "thinking entity" capable to assess facts and form judgments and the latter is a person, organization or physical entity. He identifies two main interpretations of trust, viz:

- *Reliability trust* - the perceived reliability of something or somebody; the "subjective probability by which an individual, A, expects that another individual, B, performs a given action on which its welfare depends".

- *Decision trust* - to view trust as a decision to enter into a situation of dependence.

Josang also considers *Reputation* in the context of relationships between Trustor / Trustee. He notes that trust is a personal judgment about another entity based on various factor whereas "Reputation can be considered as a collective measure of trustworthiness (in the sense of reliability) based on the referrals or ratings from members in a community". Thus, a *Trust System* produces a rating that reflects one party's subjective view of another entity's trustworthiness whereas a *Reputation System* rates an entity's trustworthiness (via a reputation score) as seen by the whole community. In this context trustworthiness can be equated to *reliability.*

Trust as a concept is also related to IT security where it can take a variety of meanings. IT computer system security mechanisms provide protection (confidentiality, availability, integrity) to data and compute resources against attacks.

Bishop defines a computer system as trustworthy if there is sufficient evidence that it meets specified security requirements [13]. A number of technologies have also been identified as key enablers of system trustworthiness. Two particular such technologies that are significant in this regard are *trusted computing* and *blockchain*. These are discussed later in this section.

Another issue to consider is reputation in Peer-to-Peer (P2P) systems. Individual peers rate transactions with other peers (trust) and these individual scores can be collected and weighted to produce a reputation score by the reputation system [14]. The amount of information collected from peers can vary according to the needs of the particular P2P system. In a small system a peer may use only its own trust ratings to rank other peers whereas in a larger system a peer may employ a *transitive trust* chain i.e. ask its "friends" (one-hop trusted peers) or "friends of a friend" (multi-hop trusted peers) right up to a *global history* reputation system that collects transaction history for all peers from all peers. Reputation scores can be calculated by centralized entities or in a decentralized manner by all the peers e.g by using an algorithm such as EigenTrust [14]. P2P reputation systems can also take actions to incentivise sharing or punish bad behaviour.

An important issue that relates to P2P trust and reputation is *anonymity*. A P2P system can typically provide various levels of anonymity from no anonymity, through some form of pseudo anonymity to full anonymity that disconnects a user's actions from his real-world identity and his other actions [14]. Full anonymity however precludes the use of a reputation system.

3.2.2 Blockchain Technology

In 2008, Satoshi Nakamoto designed Bitcoin, a peer-to-peer electronic cash system. [15] He introduced a public distributed ledger to prevent double-spending, and adopted proof-of-work to address the distributed consensus problem. His innovative work gave birth to the blockchain technology. By eliminating the trust of third parties and keeping high standard of pseudonymity, Bitcoin gained a tremendous success in business. The technology of Blockchain has also come to the attentions of researchers.

Key Characteristics

The booming of Blockchain technology in both research and industry areas is due to some unique features that Blockchain has.

- *Immutability.* Each block contains the hash link of the previous block. Due to the feature of hash algorithms, every transaction or data recorded in the Blockchain is protected by subsequent blocks. Therefore, it can never be changed. The data is secure as long as the attackers don't possess over 51% computing power. The cryptographic hash function will be detailed later.

- *Decentralization.* A common solution to verify transactions is introducing a trusted central authority. Blockchain offers a consistent public ledger to replace the central agency. To reach a consensus view of the ledger among distribute users, Blockchain employ various distributed consensus, such as Proof-of-work.

- *Anonymity.* Peers interact with the Blockchain network through generated addresses. Addresses are not directly connected to real-world identities. Furthermore, users can possess many addresses to avoid identity exposure. Information in Blockchain maintains high level of privacy. Asymmetric encryption makes it possible.

- *Auditability.* Since stored in a public tamper-proof ledger, and each transaction can be easily traced to previous transactions, this brings a high level of transparency and auditability to the Blockchain network.

In different scenarios, researchers employ Blockchain for different features. Section 3.3 surveys the use of Blockchain in the data marketplace.

Fundamental Blockchain Technology

The following sections will describe the details that make up Blockchain technology.

Cryptographic Hash Functions A hash function takes a string of any length as input, and outputs a fixed-size string, which is usually 256bits. the output string can be regard as digest of the input. A cryptographic hash function has four critical properties.

- *Collision resistance.* It is rather difficult to find two different inputs, that can produce the same output. It means one can compare two messages by comparing their hash digests which is much faster.
- *Hiding.* Given the output and hash function, it is impossible to figure out the original input. Since the message is digested to 256-bit string, there exists information loss.
- *One-way function.* It is easy to produce the output using the input. But there is no better way than brute force to find an input that can produce the same output.
- *Input sensitive.* If the input changes by even a little bit, the output will be significantly different.

Data Structure Literally, blockchain is the name of the data structure.

As illustrated in Figure 3.1, there are three basic entities within the blockchain.

- Transaction: In Bitcoin, a transaction is a transfer record. Generally speaking, a transaction is an alteration of the ledger.
- Block: A bundle of transactions and some status information compose a block.

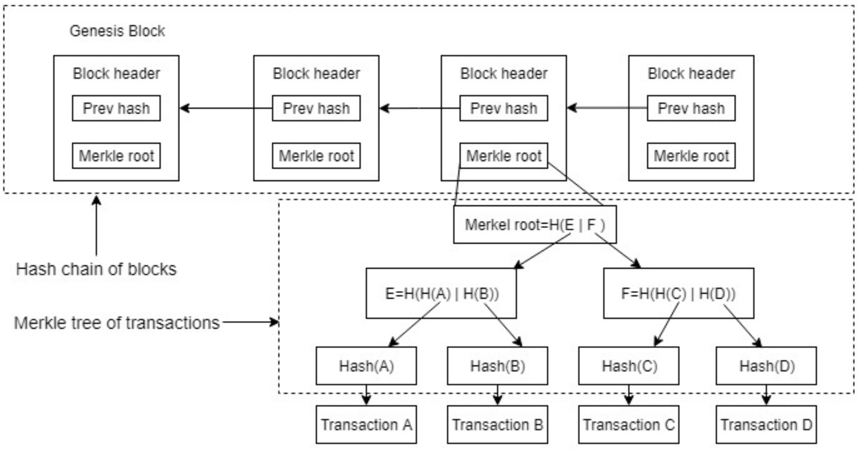


Figure 3.1: Data Structure of blockchain

- Chain: the block header contains the hash of previous block, a series of such blocks become a chain. The hash function chains all the blocks together.

The most significant advantage of blockchain structure is tamper resistance. If an attacker changes a transaction of a block, the hash of the subsequent block will not match, the discrepancy will pass to the end of the chain, i.e. all subsequent block hashes will be changed.

Merkle Tree In figure 3.1, The transactions are organized in a structure called a Merkle tree, which is also known as a hash tree. It is a binary tree with two main characteristics.

- Every leaf node stores hash of data.

- Every non-leaf node (include the middle node and the root node) contains the hash of its children's hash.

The Merkle tree is widely used in many typical scenarios.

- Quickly comparing large amounts of data

  Given that large amounts of data are sorted and stored in a Merkle tree, if the roots of two Merkle trees are the same, the two groups of data must be exactly the same.

Since the computing of the hash is very fast, using the Merkle tree to compare mass data will be of high-efficiency.

- Quickly tracing the discrepancy

  As shown in figure 3.1, if anyone changes transaction C, it will consequentially change data F and the root. By tracing $root \rightarrow F \rightarrow C$, to locate the discrepancy will be achieved in $O(\log n)$.

- Zero-knowledge proof

  Take figure 3.1 as an example, if someone wants proof that Transaction B is in the block without revealing Transaction B, by giving the value of H(A), F, he can reconstruct the Merkle tree root easily. If the reconstructed root is equal to the given root, the ownership is credible. The Merkle tree can do Zero-knowledge proof in $O(\log n)$.

The Merkle tree is essential to long-term sustainability. As more and more blocks are appended to the chain, the disk space bitcoin occupies is growing by over a gigabyte per month. The Merkle tree will help the implementation of "lightweight node". The lightweight nodes download the block headers which contains the Merkle root. It will be assured that all the transaction data is correct.

Distributed Consensus The consensus problem is one of the main challenges in any distributed system. If the data in different nodes are not consistent, the data is problematical. How to maintain a single version of the truth is a primary goal of distributed system. The basic approach is to make sure every node can read while one node can write at one time. The Paxos algorithm is a well-known consensus algorithm in a network of unreliable nodes. All the members compete as the proposer or leader - the proposer will act as a coordinator to mediate the dispute.

However, in the Bitcoin network, some nodes may crash, minority nodes may be malicious. The consensus problem in such scenarios is also known as the

Byzantine General's Problem. It cannot be addressed by the Paxos algorithm. Bitcoin applies the idea of proof-of-work as a practical solution to the Byzantine General's Problem. In bitcoin all the nodes compete for the right to append block to the ledge. The first node who solves a hash puzzle can create the block. Since the hash puzzle is rather difficult to compute, the nodes with higher computing power gain a higher probability to create the block. The longest proof-of-work chain will be the consensus of the unique ledger. The ledger is secure as long as honest nodes collectively control more computing power than any cooperating group of attacker nodes.

Proof-of-work brings two significant advantages to the bitcoin consensus. Firstly, by applying proof-of-work the network selects nodes in proportion to computing power. This progress is hard to be manipulated. Secondly, there is a cost to subvert the chain, the cost will significantly outweigh the expected profit, so it is unwise to sabotage the system.

Proof-of-Work Proof-of-Work literally means to validate nodes by the proof of workload. It was originally a countermeasure to denial of service attacks, or Sybil attacks. Bitcoin adapts proof-of-work to make the decision by the majority in proportion to their computing power.

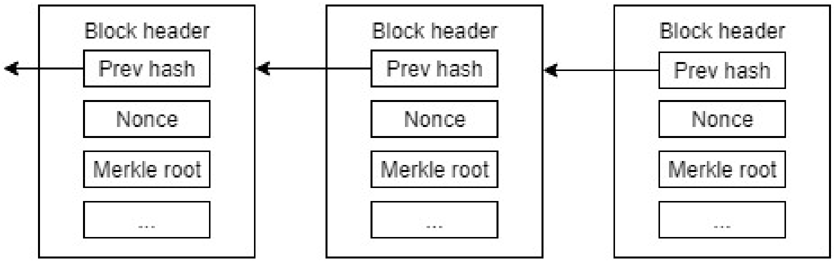


Figure 3.2: Block with nonce

As shown in Figure 3.2, Bitcoin implements the proof-of-work by adding a nonce in the block. The block hash will change along with the nonce. When the block hash has the required zero bits in the head, the hash puzzle is solved. The nodes who attempt to solve puzzle are called miners. Since the hash puzzle is difficult to compute, miners have to consume large amounts of computing resource to find the nonce. The first node to figure out the nonce will be able to create and broadcast the block. Accordingly, the miner will get the rewards.

Incidentally, by changing the number of zero bits, the system is able to adjust the difficulty of proof-of-work.

Incentives The miners consume a great deal of computing resource and electricity to find the nonce and create the block. Meanwhile, they collate new transactions and broadcast the block. It is essential to compensate the miners. In Bitcoin, miners will gain two kinds of reward from mining.

- Block reward. Every time a block is created, a certain amount of Bitcoin (BTC) will be the miner's reward. In the beginning, the block reward is 50 BTCs, the number halves every 210 thousand blocks created. Currently the block reward is 12.5 BTCs. The theoretical total BTCs is about 21 million.
- Transaction fees. If the output value of a transaction is less than its input value, the difference is a transaction fee. As time goes by, the block reward will vanish eventually, transaction fees will be the only incentive.

Smart Contract The Smart Contract is a term firstly coined by Nick Szabo in 1990s. The underlying idea is that many kinds of contractual clauses can be embedded in the hardware and software, so that any breach of contract will be expensive [16]. Besides, Smart Contracts can reduce the regulation and transaction cost, imposed by third parties. Digital Cash is a premier example in Szabo's design.

There is a strictly limited scripting language which is dedicated to facilitate the currency transaction in Bitcoin. The Bitcoin script is a Smart Contract limited to cryptocurrency. Ethereum is designed to be a general-purpose programmable blockchain. In the context of Ethereum, Smart Contracts refer to immutable computer programs that run deterministically on the decentralized Ethereum world computer. [17] Since all the nodes of Ethereum operate on the same initial state and produce the same final state, the system can be viewed as a single "world computer". More specifically, the Ethereum world computer can be considered as a single-threaded machine.

Ethereum Bitcoin is the first implementation of Blockchain Technology to prove credible in the area of cryptocurrency. Researchers have investigated how to extend the capability of Blockchain. Vitalik [18] designed Ethereum, named the second generation of Blockchain, which provides a blockchain the ability to run code. Ethereum is a blockchain with a built-in Turing-complete programming language, allowing developers to create decentralized applications. Ethereum may be seen as a general implementation of a crypto-law system [19].

In bitcoin, there is no concept of accounts. Users possess various Bitcoin addresses, but there is no balance of each addresses. The validity of a transaction is checked by the Unspent Transaction Output (UTXO) mechanism. In contrast in Ethereum, there are generally two types of accounts: Externally Owned Accounts (EOAs), and contract accounts.

- *Externally Owned Accounts.* EOAs are controlled by users, usually via a wallet application which contains the users' private keys. EOAs do not have any associated code or data storage.

- *Contract Accounts.* Unlike EOAs, contract accounts are controlled by smart contracts. Contract accounts have both associated code and data storage.

Only an EOA can initiate a transaction, and transactions can call contracts.

Ethereum Transaction In bitcoin, the transaction is used to transfer cryptocurreny among each other. It is dramatically different in the case of Ethereum. Transactions are signed messages originated by an EOA and used to change the globle state, or launch a contract to execute in the EVM.

Since a transaction is able to call various Smart Contracts, and contracts can invoke each other, infinite loops and inadvertently resource-devouring transactions bring the halting problem: there is no way to tell whether a given program will ever halt or not.

The Gas mechanism is designed to address this problem. Every computation in a contract costs gas, and the amount of gas is limited in every transaction.

There are two types of transactions: contract creation transactions and message call transactions.

- *Contract creation transactions.* Transactions will create new contracts on Ethereum for future use. They contain the compiled bytecode to create Smart Contracts.
- *Message call transactions.* They are the normal transactions which are used for the state transition or contract invocation.

These two types of transactions have some fields in common:

- *Nonce.* A number created by the sender is designed to avoid message replay.
- *Gas price.* The number of Wei to be paid per unit of gas.
- *Gas limit.* The maximum amount of gas this transaction can costs.
- *To.* The recipient Ethereum address.
- *Value.* The number of Wei to send to the destination.
- *v, r, s.* The three values of an ECDSA digital signature to determine the sender of this transaction.

Specially, a contract creation transaction has this field:

- *Init.* A byte array specifying the contract code. it will be executed once at account creation and discarded thereafter.

On the other side, a message call transaction has this field:

- *Data.* The variable-length binary data playload.

3.2.3 Trusted Computing

Along with the computer's evolution from calculators to information infrastructure, Trusted Computing become an increasingly pressing requirement.

As early as 1983, The United States Government Department of Defense (DoD) issued Trusted Computer System Evaluation Criteria (TCSEC) [20]. These criteria are divided into four divisions: D, C, B and A. Dvision A stands for the most comprehensive security. It defines Trusted Computing Base (TCB) as the totality of protection mechanisms within a system, including hardware, firmware, and software, the combination of which is responsible for enforcing a computer security policy. In order to minimize security risks, the TCB should be kept as small as possible.

Later in 1999, Compaq, HP, IBM, Intel and Microsoft established Trust Computing Platform Alliance (TCPA) [21]. As the successor to the TCPA, the Trusted Computing Group (TCG) was founded in 2003. TCG's core technologies include specifications for the Trusted Platform Module (TPM), Trusted Network Communications (TNC) and Self-Encrypting Drives (SED) [22].

TPM is a standard for a secure cryptoprocessor, which is a microcontroller dedicated to secure hardware through integrated cryptographic keys. It will be further detailed later. TNC is an architecture designed to enable posture checking, behavior monitoring, and remediation of both user endpoints and infrastructure devices. SED is a encrypt technology which enables the hard drive automatically and continuously encrypts the data on the drive without any user interaction.

With the advent of new information technologies and their various application scenarios, the concept of Trusted Computing keeps growing. There are some generally recognized features that trusted computing possesses [23]:

- *Secure boot* allows the system to boot under a defined and trusted configuration.

- *Curtained memory* will provide strong memory isolation. Memory cannot be read by other processes including operating systems and debuggers.
- *Sealed storage* allows software to keep cryptographically secure secrets.
- *Secure I/O* thwarts attacks like key-stroke loggers and screen scrapers.
- *Integrity measurement* is the ability to compute hashes of executable code, configuration data, and other system state information.
- *Remote attestation* allows a trusted device to present reliable evidence to remote parties about the software is running.

TPM

One of the most famed Trusted Computing technologies is TPM. In 2001, the TCPA releases the first version of Trusted Computing Platform specifications, it defines a subsystem which "contains an isolated computing engine whose processes can be trusted because they cannot be altered" [24]. The concept of TPM evolves continually. Microsoft states a TPM is a microchip delegated to provide basic security-related functions, primarily involving encryption keys [25]. TPM 1.2 Main Specification was published in 2003. It was incorporated into billions of devices. The emerging Internet of Things and mobile networks entail new requirements which led TCG to develop TPM 2.0 specification in 2013.

According to the latest version of TCP's specification, the TPM architecture is depicted in Figure 3.3 [26].

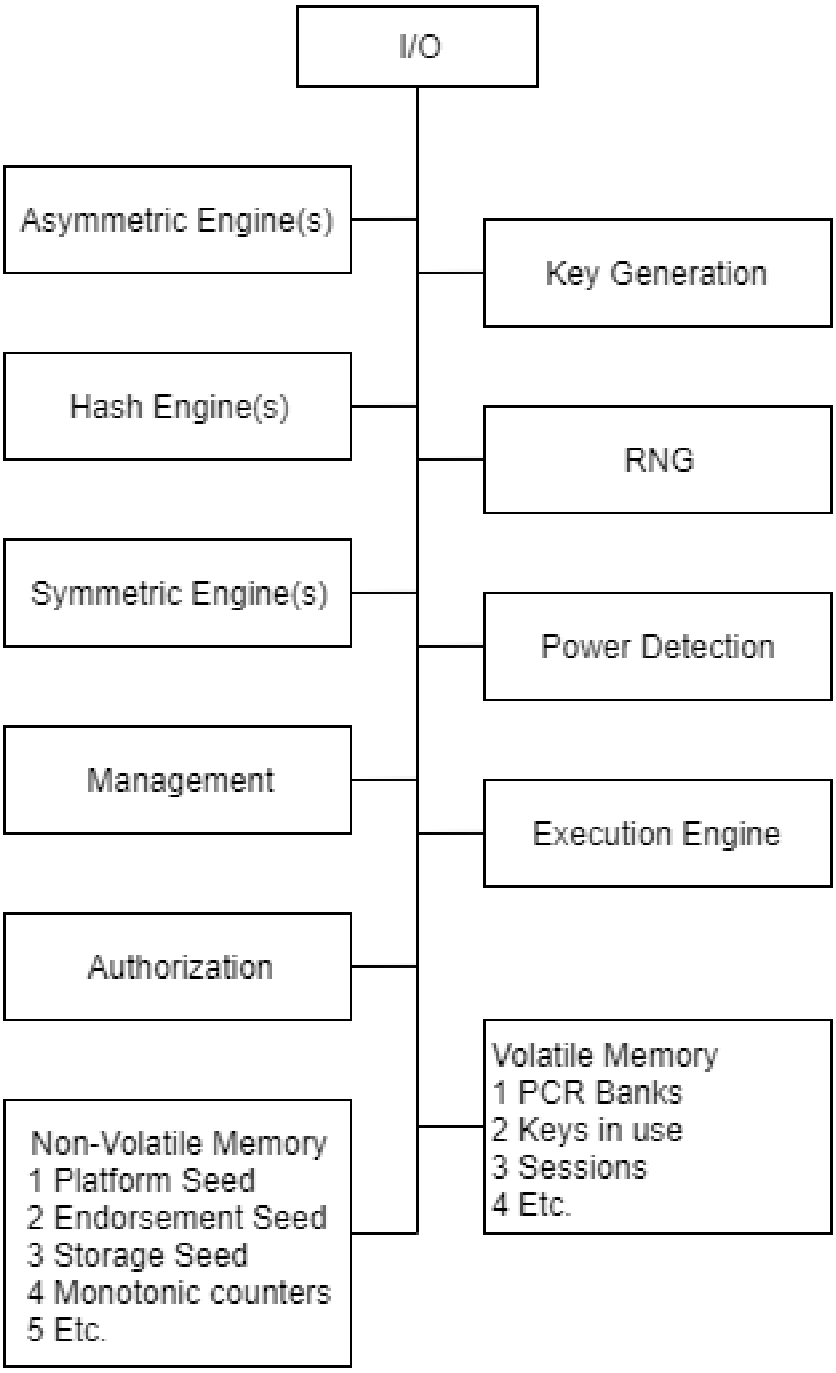


Figure 3.3: TPM Architecture

All of the TPM components are orchestrated to provide a set of predefined cryptographic services, integrity measurements, health checks and authentication services. TPM 2.0 applies a “library” approach, which allows users to choose applicable functionalities for different levels of security and implementation.

Five types of TPM are summarised by TCG in 2019 [27].

- *Discrete TPM.* A discrete chip, built and evaluated for the highest level of security, can resist all sort of sophisticated attacks. In a typical scenario, it can used to secure the brake controller in a car.

- *Integrated TPM.* An integrated chip is capable of resisting software attacks. It is separated from the main CPU.

- *Firmware TPM.* It is the code runs on the main CPU implemented to protected software. Especially, this code should be executed in so called Trusted Execution Environment (TEE). TEE will be described in next section.

- *Software TPM.* A software emulator of the TPM is suitable for testing or building a system prototype, but vulnerable in production environment.

- *Virtual TPM.* In a cloud environment, the virtual TPM is part of the cloud-based environment and it provides the same services as a physical TPM for each virtual machine respectively.

TEE

The ultimate goal of trusted computing development is the trusted application. Although TPM provides trusted cryptographic services and other related services, trusted applications on TPM are inadequate. Trusted Execution Environment (TEE) remedies this deficiency.

Before the item “TEE” was defined, various trusted environments had been implemented. Those implementations evolved from proprietary solutions to a standard approach. In 2004, Trusted Logic and Texas Instruments pioneered a generic trusted environment. 2 years later, ARM developed TrustZone, which is pervasive in ARM’s chips. In the same year, the Open Mobile Terminal Platform (OMTP) released the first set of requirements for a trusted environment, OMTP TR0 [28]. In 2008, OMTP TR1 was released, and it defined Execution Environment EE as “a set of hardware and software components providing facilities, necessary to support Applications". Any EE that meets its trust requirements is referred to as a TEE [29].

In 2010, GlobalPlatform gave their own interpretation of TEE, which is much more explicit. According to one of its latest publications, The TEE is "a secure area of the main processor of a connected device that ensures sensitive data is stored processed and protected in an isolated and trusted environment [30]."

The initial designs of the GlobalPlatform TEE and other TEEs were tailored for the mobile space, especially for secure video and audio services or secure payments. However, trusted execution is desirable virtue of every computing device. It is becoming critical in IoT, clouding computing etc. area.

A lot of chip design and manufacture companies propose different implementations of TEE. A list of them is below:

- AMD:
    - AMD Platform Security Processor (PSP)
- ARM:
    - TrustZone
- IBM:
    - Secure Service Container (SSC)
- Intel:
    - Software Guard Extensions (SGX)
- RISC-V
    - MultiZone$^{\text{TM}}$ Trusted Execution Environment

Intel SGX, dominant in the area of servers and PCs, is the technology we adapt.

In 2013, researchers from Intel published several papers to introduce the model and mechanisms of Intel SGX [31, 32, 33]. In 2015, Intel SGX began to be implemented in 6th Generation Intel Core. Mckeen et al. describe Intel SGX as the technology enabling application to instantiate a protected container, referred to as an enclave. The enclave is opaque to the operating system, thus guarantee the confidentiality and integrity of the application [31].

It supports three main activities to establish trust [34]:

1. Measurement - provides an accurate and protected recording which measures and proves the identity and integrity of an enclave.

2. Attestation - allows an enclave to prove its identity and authenticity to another party. Attestation includes both local and remote attestation.

3. Sealing - ensures that the data is only revealed in the trusted environment, and it is encrypted when stored outside this environment.

Attestation

Attestation is a mechanism that a service provider can prove its identity to the service consumer. A claimed trusted computing service may run in an emulator, as show as Figure 3.4. Attestation is the crucial part to solve this sort of concern.

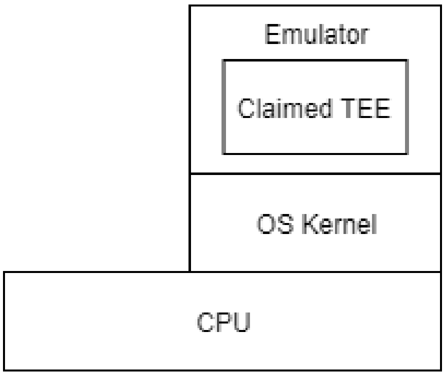


Figure 3.4: Why is attestation needed?

Both TPM and TEE has their attestation implementations, their basic idea is in common:

- Each trusted computing chip contains a pair of keys which is concealed and certified by the Certification Authority (CA).
- The provider generates a report, i.e. attestation data, through the private key.
- The verifier, or challenger will verify the signatures and looks up the report in a trusted database.

The following picture illustrate the basic attestation protocol in TPM [23]:

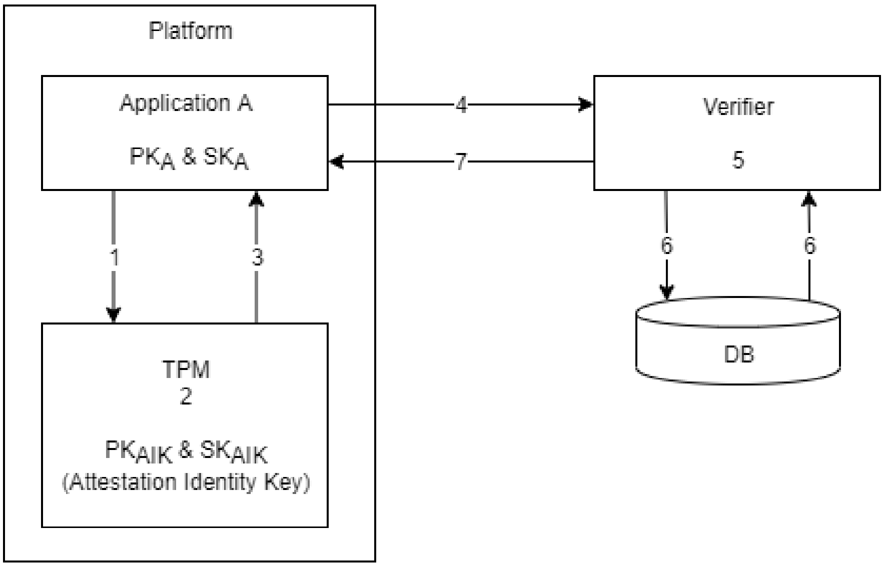


Figure 3.5: Attestation protocol

1. The application "A" generates a pair of keys: PK_A & SK_A and send PK_A to the TPM.
2. The TPM computes a hash value #A of the executable code of "A".
3. The TPM creates a certification including PK_A and #A and signs it with SK_{AIK}
4. "A" sends the certification along with the cert. of PK_{AIK} to a verifier.
5. The verifier verifies the cert. chain,
6. then look #A up in a trusted database.

7. If “A” is proved trustworthy, they will establish a session key using PK_A

The attestation of TPM contains one critical limitation. Attestation happens at the time executable code is loaded and is therefore unable to provide a view into program behavior at run-time [23]. A program may be compromised during execution, which can't be detected by attestation.

Differently, in a TEE the program code resides in isolated secure environment, which is invisible to attackers. The attestation of TEE preferably proves the application is trustworthy.

3.3 Data Sharing Using Blockchain

Blockchain can be utilized as a public distributed consistent database, which can be adapted to share information. The follow sections surveys what benefits Blockchain brings to their system, and how Blockchain is implemented to accommodate their needs.

3.3.1 IoT Data

The existing Internet of Things (IoT) data structure is cloud-centric architecture. It causes a myriad of isolated data silos, which limits full use of holistic data-driven analysis. Hossein et al. [35] adapt Blockchain as an access control system and empower the users with data ownership. The raw IoT data is stored off-chain, compatible with both centralized and decenrtalized file systems. The Blockchain system is adapted to manage the ownership and access permissions of data by third parties.

Facing the same problem, Kazim et al. propose a blockchain-based, decentralized and trustless data marketplace, IDMoB, where IoT devices vendors and data consumers may interact and collaborate. [36] End users may be able to use IoT devices and services freely in exchange for their consent on data usage. The consented data in stored in Swarm, a distributed storage system. The researchers argue that free-to-use or free-if-consented business models bloom in every industry. Since the data is stored and managed by a distributed system, there is no way for anyone to monopolize the datasets. In this system, data is also stored off-chain. Transactions are used to manipulate data, such as upload, query, request.

3.3.2 Personal Data

There is a growing public concern about user privacy. Centralized organizations amass large quantities of personal and sensitive information. Researchers propose a decentralized personal data management system to ensure users ownership. Guy et al. [37] re-purpose a blockchain as an access-control moderator, and design two types of transactions: one is used for access control management; the other is used for data storage and retrieval. This system requires a network of nodes or a centralized cloud to store the off-chain data.

3.3.3 Health Information

Hospitals, clinics, institutions, and IoT Devices will produce various healthcare data, using varying structures and terminologies and vocabularies. These stucture and semantics issue will preclude compatibility, thus limit the utility of the data. Kevin et al. [38] utilise Blockchain to share health information, and the terms of data access is strictly dicated by the patient. They introduce a new consensus algorithm to facilitate data interoperability. Transactions, representing patient record, do not inclute the actual record documents, but Uniform Resource Locators (URLs) of those documents.

Ariel et al. [39] design a health information sharing platform, which use Smart Contracts to orchestrate a content-access system across separate storage and provider sites. The contracts contain metadata about the record ownership, permissions and data integrity, while the transactions carry cryptographically signed instructions to manage these properties.

3.3.4 Discussion

Blockchain Technology is widely used in data sharing. The reason why researchers adapted Blockchain to share information can be summarized as follows:

1. To give users control of their data.

   There is no central corporations or organizations that can control or manipulate big amount of information. Only allowed third party can access users' data.

   In Shafagh's work [35], transactions convey access permissions, users can assign or revoke permissions by sending transactions. Zyskin et al. [37] use the same idea to protect personal data. They create two types of Transactions: Transaction of access control and Transaction of data. "MedRec" [39] use various Smart Contracts to record Patient-Provider Relationship.

2. To motivate data sharing by various incentives.

   In "IDMoB" [36], users can trade their consent on data usage with free-use of Iot devices and services. IoT manufacturers can benefit from data monetization. "MedRec" [39] encourage participantion by granting patients a holistic, transparent health record, and providing researchers relevant clinical data.

3. Decentralized marketplace.

   In all the systems above, peers can directly share information with each other without a third agency, which will reduce the transaction cost and lower the barriers to entry, which will encourage people to participate.

4. Integrity of data.

   Because of the feature of Blockchain, all the data in Blockchain is immutable and transparent, and all the data stored off-chain is protected by the cryptographic hash function. Thus, the data is trustworthy.

At the system-implementation level, there are three major concerns within every Blockchain system:

- How is the data stored?
- What does the transaction do?
- Does the Blockchain performance meet system requirements?

As to the first question, all the previous works store the raw data off-chain, and keep metadata on-chain, such as Hash digest, location data and so on. Shafagh [35] designs the Data plane which is agnostic to different types of storage. "IDMoB" [36] stores the data in Swarm, a peer-to-peer storage platform. "MedRec" keeps the healthcare record in the local database of each node.

As regards the second question, they have dramatically different design. Shafagh [35] use transactions to accommodate access permissions. Zyskind [37] designed two type of transactions, one is used for access control management, the other is for data storage and retrieval. Both "IDMoB" [36] and "MedRec" [39] use transaction to invoke Smart Contracts, and the Smart Contract is dedicated to register the user or manipulate the data, etc.

The key performance indicator of Blockchain implementation is Transactions Per Second (TPS). Theoretically the TPS of Bitcoin, Ethereum are 7, 15 respectively. Many implementations are built on the top of these platforms [35, 39]. The Consortium Blockchain adopt various high-performance distributed consensus algorithms and their TPS is much higher. According to Androulaki's test, Hyperledger Fabric achieves more than 3500 TPS in certain popular deployment configurations [40]. The balance between openness and performance is a design consideration.

3.4 Threat Intelligence Sharing

Information systems are a vital infrastructure of corporations, organizations and countries. Compromise of digital systems through cyber attack means great danger and loss to individuals and communities. For example, WannaCry, a ransomware with worm functionalities, attacked more than 230,000 computers, caused an estimation of losses between one and four billion dollars [41].

Cyber threat intelligence is the information to help organizations identify, assess, monitor, and respond to cyber attacks including indicators, Tactics, Techniques and Procedures (TTPs), security alerts, threat intelligence reports and actionable advice [6].

Threat Intelligence Sharing is an approach of exchanging cyber threat intelligence within a community, to make full use existing information and defend attacks together.

3.4.1 Benefits

There are several works that state the benefits of threat intelligence sharing. Table 3.1 is the summary.

Table 3.1: Main Benefits of Threat Intelligence Sharing

Index of Papers Items	[1]	[3]	[2]	[6]	[42]	[7]
A Better Situational Awareness	*	*	*	*		
Greater Defensive Agility	*			*		
Reduced Cost	*		*		*	*
Higher Quality of Knowledge	*	*		*		*

- *Better Situational Awareness.* Participants may collect a great deal of external threat intelligence, which will grant them a better situational awareness of the threat landscape. A single indicator of compromise could increase the awareness of a whole community. Sharing information will make full use of a threat intelligence [1, 3, 2, 6].

- *Greater Defensive Agility.* Attacks usually try various Tactics, Techniques, and Procedures (TTPs) to evade detection and exploit vulnerabilities. Sharing information will identify the changing TTPs faster. Information sharing can speed identification and detection of threats, which will prevent potential or ongoing attacks [1, 6].

- *Reducing Costs.* In general, cost-saving may stem from quicker reactions to attacks. Sharing information may help members avoid the denial of services attacks or other attacks. Secondly when an organization finds it has been under attack, it usually immediately invests time and money to develop a countermeasure. If another organization is under the same attack, they could develop defense measures together and share the cost. Besides, the adoption of automated exchange mechanisms may reduce human resource costs, and increase scalability [1, 2, 42, 7].

- *Higher Quality of Knowledge.* Sharing information would connect data silos. When seemingly unrelated observations are exchanged, that information can be correlated with data collected by others. The correlation process will increase the value of isolated information by discovering implicit relation among data set [1, 3, 6, 7].

3.4.2 Challenges

Threat intelligence sharing has significant advantages, while challenges still remain. Technical challenges are as follows:

Table 3.2: Main Technical challenges of Threat Intelligence Sharing

Index of Papers Items	[1]	[3]	[2]	[6]	[42]	[7]
Trust Issues	*	*	*	*		
Privacy Issues	*	*		*		*
Quality Issues	*	*	*	*		*
Interoperability Issues		*	*	*		
Hierarchy Architecture		*		*		
Incentive For Sharing					*	*
Negative Publicity	*				*	*

- *Trust issues.* There are three types of trust that need to be taken into consideration. Firstly, trust between information consumer and information provider is of vital importance. Organizations will only share sensitive information with organizations they trust. Secondly, all participants must trust the threat intelligence sharing platform, otherwise they will not participate in such a platform [1, 3, 2, 6]. Thirdly, information consumers need to trust the quality of the sharing information.

- *Privacy issues.* In terms of sharing information across organizations, privacy information contains both Personally Identifiable Information (PII) and organization secrets. PII refers to information which can be used to distinguish or trace an individual's identity, such as name, social security number, biometric records. To recognize and safeguard PII is one of the main challenges. Organization secrets include intellectual property, trade secrets and other proprietary information. Disclosure of this privacy information may cause violation of legal rules or financial loss [1, 3, 6, 7].

- *Quality issues.* Threat intelligence quality includes relevance, timeliness, accuracy, comparability, coherence and clarity. Before acting on received information, organizations need to confirm that the information is both correct and relevant. So, evaluating the quality of threat information is required [1, 3, 2, 6, 7].

- *Interoperability issues.* Organizations often employ their own terminology and data standards, which limit interoperability and automate data processing between members. Standardized data model, data formats and transport protocols are required for interoperability. Structured Threat Information eXpression (STIX) is the most commonly used standard for describing threat intelligence [3, 2, 6].

- *Hierarchy architecture.* Different communities share different levels of classified information. For example, normal citizens are not permitted access to government classified information. An organization may share general threat intelligence to public, sensitive information to community, and restricted information to participants they trust [3, 6].

- *Incentive for sharing.* Since information sharing may cause privacy disclosure and financial loss, participants are reluctant to disseminate their information. Some free-riders may consume valuable intelligence without sharing information with equal value. Thus, creating incentives is indispensable to sharing [42, 7].

- *Negative publicity.* Some sensitive information should be restricted internally or among few specific participants. Negative publicity might affect organization's market value and stock price. Competitors might use the information against the informant [1, 42, 7].

3.4.3 Requirements for TIS system

An adequate and efficient TIS system should address all the above challenges. To meet the privacy and legal requirements, Personal Identifiable Information (PII) and organizations secrets should be masked, anonymized or removed. For the sake of trust issues, encryption techniques, digital signatures, attestation methods, access control system, Blockchain and Trusted Computing, Smart Contract, and so on, are available partial solutions.

Besides those technical barriers, various non-technical issues also need to be considered, such as policies governing membership schemes [3, 43]. Many data sharing tools do not fully cover the needs of the communities. Serrano summarises the minimal requirements of any cyber security data sharing system, and proposes an "Information Exchange Policy (IEP)" to govern all exchange of data [3]. The IEPs stipulate at least the following components:

1. IEP related elements:

    - Purpose of the exchange covered by the IEP

- Agreed procedure for modifying the IEP
- Scope of the information covered by the agreement

2. Participant related elements:
   - List of participants
   - Agreed procedure for admitting new participants
   - Instructions for handling received data for leaving participants
   - Whether or not exchanged data can be modified by recipients and subsequently forwarded
   - Uses that can be made of exchanged data

3. Data related elements:
   - Verifying the quality of exchanged data
   - Agreement on the minimum quality of exchanged data
   - Mechanisms and communication channels to be used for exchanging data
   - Intellectual property rights of the exchanged data, covering the following:

     - Reproduction

     - Distribution / public performance/ broadcasting

     - Commercialization

     - Moral rights
   - Agreed retention procedures of the exchanged data
   - Anonymization required for exchanged data.

Most of the data related elements can be encoded within the TIP, while others need to be negotiated within different communities and enforced through audit and other channels.

On Serrano's proposal, he did not predict the trend from the central solution towards a more decentralized, dynamic based model, which demands more consideration on open trust issues.

Trust Requirements in TIS

Although sharing of threat intelligence is happening, its effectiveness is greatly held back due to trust related issues [1, 3, 2, 6, 7]. In this section I consider further the three trust concerns raised earlier with the goal to identify requirements for the proposed TITAN system. These issues are:

Trust between Participants This type of trust is of paramount importance and is related to the type of TI, degree of sensitivity of the TI data and the purpose for which it is exchanged. Sensitive data that could have a negative impact on a business will be shared only with most trusted partners. Such sharing is often governed through use of the Traffic Light Protocol (TLP) protocol which defines categories of sensitivity and corresponding principles for sharing [44]. Higher TLP category Information is more likely to be of a coarse granularity and very often is shared via email or through direct (either physical or electronic) conversations.

Participant trust also relates more generally to sharing communities. Tounsi reports that trust is lowered when some community members are seen to be under-contributing i.e. freeloading [1]. This is the classic "selfish" peer in P2P network literature [14]. Furthermore, the potential exists for malicious peers in P2P networks who may disseminate false or poor-quality information with the aim to disrupt or harm the operation of the community. This aspect of trust is likely to become more critical as TI sharing evolve towards a more market-based model.

Trust as discussed here relates very much to the notion of "Reliability Trust" discussed earlier. The relationships may be subtly different for the two cases described here however. In general, in the case of TLP based sharing the Trustor is an *information publishing* entity whereas in the broader community-based sharing scenario the Trustor is the *information receiving* entity. In the first case trust is gained over time from ongoing contacts whereas in the second case participants will need a mechanism to estimate the trustworthiness of their counterparty. Reputation systems have been suggested as a means of to achieve trustworthiness and incentivise TI sharing [1], and, more generally, reputation systems are seen as very effective means of incentivising information sharing in P2P systems [14].

Trust in the TI Platform This aspect of trust relates to the notion of IT security trust discussed in the previous section. In general, many different mechanisms will be used to ensure that data confidentiality and integrity is maintained including strong access control, encryption of data at rest and in transit, VPN and so on. Platform trust also helps to enable participant trust particularly for communities as many of the policies governing community participation are likely to be directly supported by platform security mechanism such as those mentioned here.

The trust technology enablers such as Blockchain and Trusted Computing are of course also contributors to trust in a sharing platform. To-date these technologies have not been widely deployed and I discuss their role further in my own design in the next chapter.

Trust of TI Quality The quality of threat intelligence is based on attributes such as relevance, timeliness, accuracy, comparability, coherence and clarity and provenance [7]. Threat intelligence quality (TIQ) trust is becoming evermore critical as the volumes of threat intelligence grow rapidly larger and threaten to swamp cybersecurity team's capability to process incoming information [7, 3, 45, 46]. Having a high degree of trust in the reliability of the information quality is therefore a very important factor in deciding which TI data or data source, or community, to join with.

As TI sharing architectures move toward a more dynamic, decentralized form it will become increasingly more difficult to judge information quality [45, 8]. This will be exacerbated by the problem of assessing quality across different dimensionalities and the fact that threat information can be of many different forms. Furthermore, the mode of delivery is likely to influence assessment of quality e.g. assessing the quality of a synchronous stream of security events versus more asynchronous form of information e.g. incident reports or event data files may require different mechanisms. Sillaber conducted a focus group with security experts to determine the challenges of assessing TIQ [45]. His recommendations for future TI sharing practise include the need

1. to inform users of the TIQ since trust in data is of the utmost importance.
2. to crowdsource TIQ management i.e. to allow a publisher's subjective quality to be ranked by other participants via a reputation system.
3. to automate the TIQ assessment process. This is required due to the sheer volume of information shared.

Al-Ibrahim [8] attempts to measure the effectiveness of TI sharing by assessing the quality of the shared data versus the volume of shared information - the more traditional approach. He introduces the notion of *Quality of Indicators (QoI)* to assess threat intelligence along various quality dimensions. He posits that the concept will help TI improve including amelioration of the free-riding problem.

Future Threat Intelligence Platforms will need to support a variety of data quality assessment forms such as computational trust [47] or machine learning [8]. Furthermore, these systems are likely to allow assessment be provided by either publisher, consumer or a third-party service [8]. In order to support this flexibility, it will be necessary for the TIP to be able to independently verify the integrity of operation of the assessment process in case of dispute. An added complexity for assessment system design will be the need to deal with a variety of

TI sharing forms including streaming alerts, security alert files, incident reports and other discrete data.

3.4.4 Architecture for TIS system

Peers share information in various ways, emails, chats, group meetings, ftp servers, P2P file sharing systems etc. On the conceptual level, all of those methods can be characterized into two models: P2P and hub-and-spoke [43]. Some systems clearly stand at one side, while other employ blended approaches.

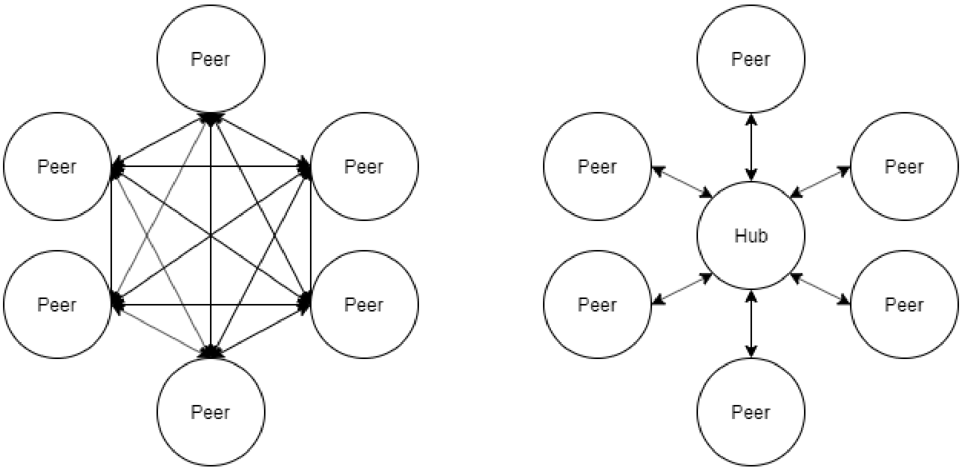


Figure 3.6: TIS Architechtures

As depicted in Figure 3.6, the key difference between two architectures is whether a central hub exists or not. The comparison between two architectures is represented in Table 3.3.

Peer-to-Peer

In a pure P2P model, each participant, equipped with equal capabilities, can serve as both a server and a client. Without any "gatekeepers", peers can join in and leave off the community at their will. This architecture functions well as long as there is more than one participant. And they can share information directly without any intermediate link, thus without any censoring or filtration from third parties. As a consequence, P2P model is especially beneficial for smaller communities or sharing with the minority of a community [43].

Table 3.3: Comparison of TIS Sharing Models

	Peer-to-Peer	Hub-and-Spoke
Actors	Peers	Peers and a hub
Advantages	• Without trust for central authority • Dynamic community • High robusticity • Less latency • Natural load balancing • High efficiency when sharing in small community.	• Easy to deploy • Easy to enforce various policies • High efficiency when broadcasing
Challenges	• Redundant sharing • Hard to supervise	• Requirement for the central trust • High dependency on the hub • Single point failure

Since there is no central authority in this model, supervision over the community is an emergent challenge, which can be partly mitigate by Smart Contracts. Another consequence is that redundant sharing on the same information happens more frequently in this model.

Hub-and-Spoke

Hub-and-Spoke architecture is much more widely used in industry area. On the contrary to P2P model, a gatekeeper is established at the center of the community, i.e. a hub. All the participants offload some functionalities to the central agent. In addition to that, the hub is usually furnished with management and regulatory capacities. Take this as the background, Hub-and-Spoke models have a much wider range of technical options. It is easier to deploy and to enforce various policies.

Benefits and challenges are the two sides of the same coin. If the hub does not perform as well as it should, the whole sharing community is in paralysis or in danger. A peer only joins in the community if the hub is trustworthy. All the regulation data and sharing data is funneled through the hub. It makes the hub a performance bottleneck. Therefore, severe dependency on the hub leads to a high possibility of the Single point of failure.

Hybrid Approach

As discussed above, both architectures have valuable advantages along with significant limitations. A hybrid approach combining elements of both would hedge the risks and extend the strengths.

The incorporation of different models should take two factors into consideration:

- Functionality. A TIS may contain diverse services or functionalities. Some services involve multiple parties, such as threat intelligence correlation and analysis. The services which cannot be completed by a single peer is more likely to be host a Hub-and-Spoke module. Conversely, some services which can be executed on the edge network is compatible with P2P models.

- Scope of Sharing. Some kind of threat intelligence is of high confidentiality, which can be only shared with a small group. The less parties involved the better. In this case the P2P model is a suitable choice. On the other side if some general information needs to be broadcast as widely as possible then Hub-and-Spoke is a more efficient option.

3.4.5 Knowledge Exchange

There are multiple sharing communities residing in the same platform. Different information is shared within specific organizations or communities, which requires a hierarchy sharing architecture.

To address the above demand, Serrano proposes the concept of Knowledge Exchanges (KEs) [3]. A KE is a service which maintains a list of data sharing organizations and their associated data and/or service offerings. The listed services may include, but are not limited to, data anonymization, data evaluation, data translation, data correlation, GDPR verification. Organizations can access data and request services through KEs, they may interact with data and services providers directly. A KE can be initialized by any participants, then organizations may join in different KEs according to their needs and publish their data and service offerings on existing KEs. KEs are independent to the information sharing system itself.

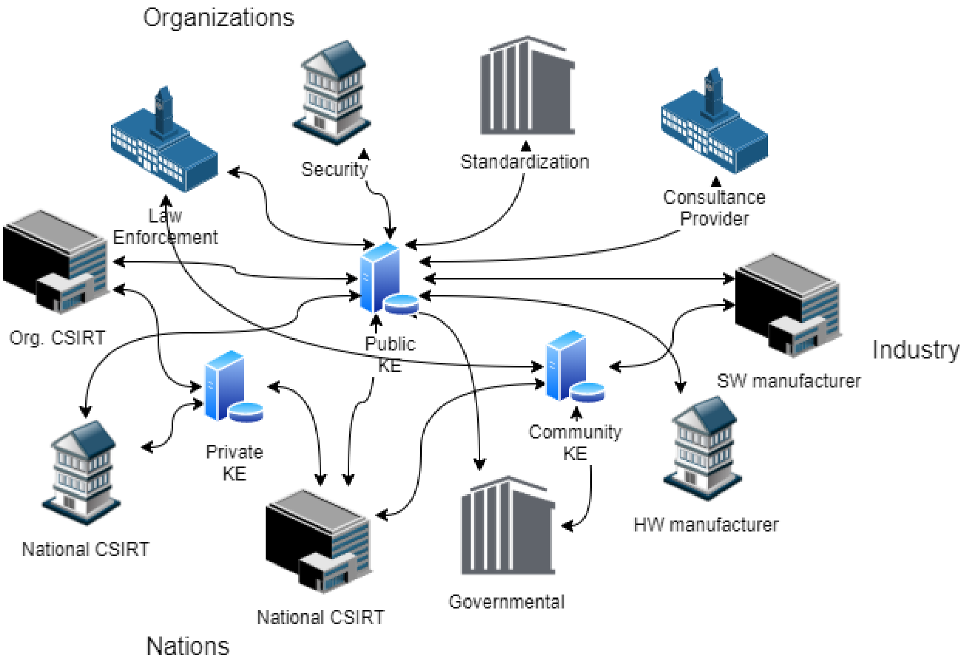


Figure 3.7: Concept of Knowledge Exchanges

Figure 3.7 illustrates three main types of KEs:

- *Private.* These KEs are isolated from other organizations and KEs. They adopt strict authentication and authorization processes to ensure membership through trusted channels. They are suitable to share highly sensitive information within small groups.

- *Community.* Community KEs also implement strict mechanisms for authentication and authorization. Private KEs are invisible to non-members. In contrast, Community KEs will exchange information with non-members based on their authentication information.

- *Public.* Public KEs, without membership schemes, are used to share public information. They maintain the list of every peers.

The concept of KE is intend to regulate the scope of sharing, but the capabilities of KEs are not restricted to this. IEP can be partly implemented through KEs.

3.5 Blockchain in Threat Intelligence Sharing

From the discussion above, I conclude that Blockchain has several major intrinsic advantages in the area of data sharing. Here I explain how I adapt Blockchain to mitigate the challenges of threat intelligence sharing I enumerate in section 3.4.

- *Trust issues.* In a vulnerable network, members are reluctant to trust each other when they are unfamiliar with each other. Blockchain gives a single public distributed tamper-proof ledger, which is trustworthy. The consensus of this ledger can become the basis of trust that underpins data sharing.

- *Hierarchy architecture.* Different communities share different levels of classified information. This can be addressed by an access control system. Hossein [35], Guy [37] have designed fine grained access control system. Besides, Smart Contracts grant Blockchain the ability of sophisticated computing that can be used to identify different communities and different levels of information. Thus, Smart Contracts is capable of supporting advanced hierarchy architecture.

- *Negative publicity.* A hierarchy architecture brings fine granularity of information designation and the scope of information dissemination is strictly restricted. This will reduce negative publicity.

- *Incentive for sharing.* An incentive mechanism is one of the reasons contributing to Blockchain's success. By creating the block, creators will gain block reward and transaction fees. With economic incentives and violation cost, most nodes behave as expected. In threat intelligence sharing organizations are reluctant to share information due to the fear of privacy disclosure and financial loss. So, an effective incentive mechanism is the more pressing requirement for our system.

Although Blockchain helps a lot to mitigate those challenges, there are some challenges Blockchain can not address directly.

- *Privacy issues.* PII and organization secrets need to be removed or masked before information sharing can occur. Blockchain won't help because it works on the sharing procedure. IEP will help organizations to execute the preprocess procedure, such as pattern matching, extraction, and obfuscation. Blockchain will use cryptography to protect every information on the ledger and can adopt an audit system for forensic analysis.

- *Quality issues.* Blockchain will not directly improve the quality of threat intelligence. A data quality assessment module will give assistance to improve data quality. Information that does not reach the required quality criteria will not be validated nor accepted. To that extent, blockchain can assist improving the average quality to some extent. The information quality assessment is also a preprocessing procedure. Threat intelligence can be qualified by its completeness and freshness. Completeness is reduced when missing a field which is stipulated in a sharing data standard. Freshness will decay as time passes since detection time.

- *Interoperability issues.* The barriers to interoperability are different terminology and data standards. Blockchain will not directly analyze the information's semantics just as information's quality. A preprocessing procedure may help.

3.6 Trusted Computing in Threat Intelligence Sharing

Trusted Computing (TC) is widely used in payment, biometric and cloud computing area. Few works adopt Trusted Computing to Threat Intelligence Sharing.

In 2006 Sandhu proposed a PEI models to share information using TC [48]. As I discussed in Section 3.2.3, TC provides cryptography services capable of sealing sensitive data in the trusted environment. Based on this feature of TC, the PEI models designed an access control system which makes sure the protected content is only be obtained by certain authorized clients. In 2008 Lu presented a P2P resource sharing scheme, RS-UCON, using TC [49]. Besides sealed storage, RS-UCON exploits an attestation mechanism of TC to establish P2P communication.

All the above implementations are built on the top of TPM, which provides attestation and cryptogrphic services. But they fall short of creating trusted applications. With the advent of TEE, user-defined code can be deployed in a TC platform. TC's scope of application is widely extended.

The following challenges of TIS is effectively overcome by TC:

- Trust issues. Trustworthiness is the core value of TC. With the help of TC, the sharing process is anti-hacking, and the sharing data can be securely sealed. As a result, the trust towards the platform is greatly enhanced.
- Privacy Issues. All the privacy with in TC is invisible to outside applications. PII and organization secrets can be securely removed or masked.
- Quality issues. The data quality assessment module can be implemented in TC; thus, the quality value is tamper-proof.

All the system modules can be migrated from a normal environment to a TC environment. It will enforce the service logic to be executed as desired.

The shortcomings of a TC implementation are the hardware cost and fee due to attestation services. This is beyond the scope of my research.

Chapter 4

TITAN Design

4.1 Introduction

Our approach to address the above requirements is to develop a system to improve participant trust and data quality assessment issues using trusted platform technologies, namely blockchain and TEE. In this chapter, I explain the proposed system – Threat Intelligence sharing Trust enhANcement (TITAN).

Specifically, I propose:

1. To develop a TIQ assessment framework based on the use of TEE. This will enable trust by verifying the integrity of the received TI and the assessment processing.

2. To develop a reputation system that will allow peers to rate TI sharing transactions based on the quality of the TI received and also, more generally, to allow user subjective feedback on a range of peer transaction types e.g. to rate the quality of a different services (providers).

TITAN is aimed to be a general solution for trusted TI sharing for use across different TIPs and to incorporate a variety of TIQ assessment and scoring algorithms. The initial development of the tool is based on the use of EigenTrust and the PROTECTIVE [47] sharing TIP.

4.2 Architecture

The overview of our system architecture is depicted in Figure 4.1. It consists of three layers:

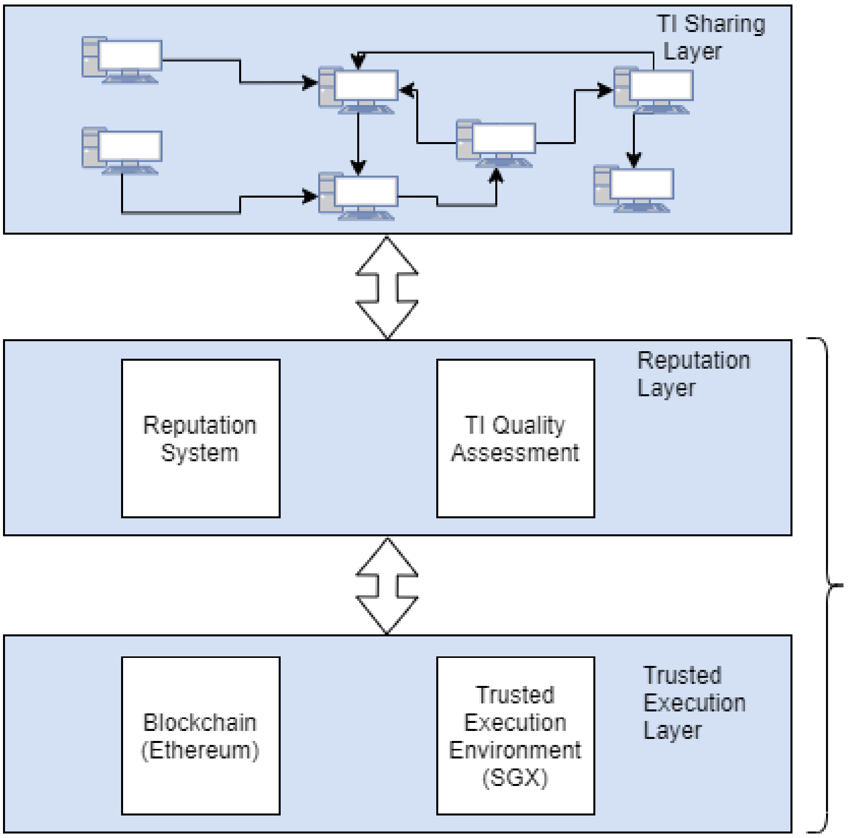


Figure 4.1: TITAN Architecture

- *TI Sharing Layer.* This is the (P2P logically connected) threat intelligence sharing community. This layer is in fact outside of our framework and instead consumes the TITAN trust services. TITAN is in general agnostic to the details of particular TIP e.g. such as underlying communication distribution models i.e. P2P, hub and spoke or hybrid, but TITAN can be adapted to TI sharing community specific assessment requirements such as TIQ assessment algorithms.

- *Reputation Layer.* The Reputation Layer is the core functionality of the system. It consists of two principal components the *Reputation System* and *TI Quality Assessment Calculation.*

The reputation system is based on the use of *global history* P2P reputation algorithms such as EigenTrust. It is primarily implemented in the Trusted Execution layer in both the blockchain and TEE components. The blockchain is composed of the following smart contract functions:

1. *User Administration* - registers and deregister peers to TITAN via *join* and *leave* functions
2. *TIQ Assesment* - responsible for managing the TI quality assessment scores via the *updateQualityScore* and *getQualityScore* functions. This contract automatically invokes the Trust Scoring contract to convert TI quality score to local trust score.
3. *Trust Scoring* - the contract updates the reputation score for a single peer interaction or transaction via the *updateTrustScore* function and the score is retrieved using the *getTrustScore* function. The trust score is defined separately from the TI quality assessment as it enables a more general trust scoring scheme. Thus, reputation may be based on i) just the TI quality score ii) a combination of TI quality scoring and other parameters or iii) without the TI quality score.
4. *Reputation Scoring* - the reputation algorithm triggers the calculation of the global trust i.e. the reputation score via the *calculateRep* function and the score may be retrieved via the *getTrustScore* function. The actual scoring calculation is offloaded to the TEE for performance reasons.

Note that the remote attestation features of the TEE means use of a third party to provide the assessment service does not change the decentralized nature of the architecture as any peer may provide the function.

- *Trusted Execution Layer.* This layer comprises the blockchain and the TEE. The blockchain stores the trust and reputation score data and provides security and integrity against manipulation of the scores. The TEE provides security and integrity for the TI quality scoring algorithm and data and prevents manipulation or tampering. The algorithm code is published so that any person can examine it and verify its operation. The input data to the assessment is cached in the TEE layer for some policy defined period to enable remote attestation in case of dispute. The reputation scoring algorithm is similarly protected.

4.3 Workflow

Figure 4.2 gives a high-level overview of the operation of the TITAN system, shown in the phases A, B and C. The figure depicts two members of the TI sharing community, containing the blockchain and TEE components.

In phase A, members of the TI sharing community register as part of the TITAN via invoking the *join* smart contract function. At this stage member accounts and other related data entities are created in the Ethereum blockchain. Participants are now able to take part in the TITAN sharing scheme.

At some later time, in phase B, one participant receives threat intelligence from a peer - how this is done is specific to a particular TIP and is not considered here. The receiving peer then requests the TIQ assessor to assess the quality of the received TI. The result of the assessment is then posted to the blockchain via the *updateQualityScore* smart contract function. After that the receiving peer or other peers can get the transaction quality score via *getTIQuality* and their own Ethereum Agents.

In phase C, a participant requests Reputation Algorithm Calculator to refresh the reputation scores by calculating the peer global trust values i.e. *calculateRep*. This will call *offload* smart contract function to retrieve all aggregated transaction scores, which will be fed into the actual reputation calculation function. Then it produces and stores the reputation results into smart contract. A peer may then, or at any time, retrieve the reputation score for any other peer via *getRepScore*

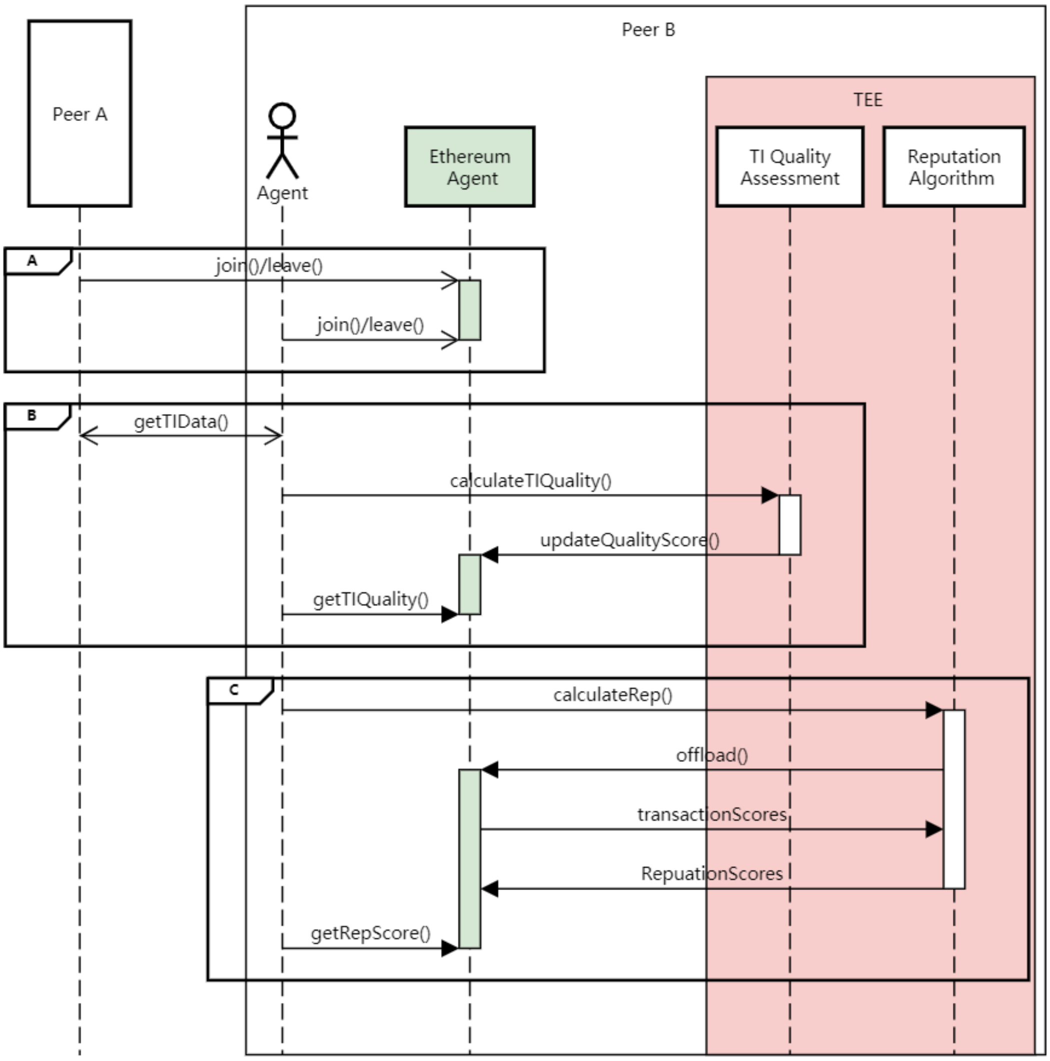


Figure 4.2: TITAN Workflow

smart contract function.

The pseudocode of the Smart Contract is shown in Appendix Algorithm 2.

4.4 Quality Assessment

TITAN is using the the H2020 PROTECTIVE TIP [47] as the TI sharing layer to develop to prototype and validate the approach. PROTECTIVE aims to improve security situational awareness through TI sharing, amongst other things. The TIP shares network and email indicators of compromise using the Intrusion Detection Extensible Alert (IDEA) schema [50].

The project has developed a TIQ assessment algorithm based on computational trust [47]. The following figure illustrates the process of assessment.

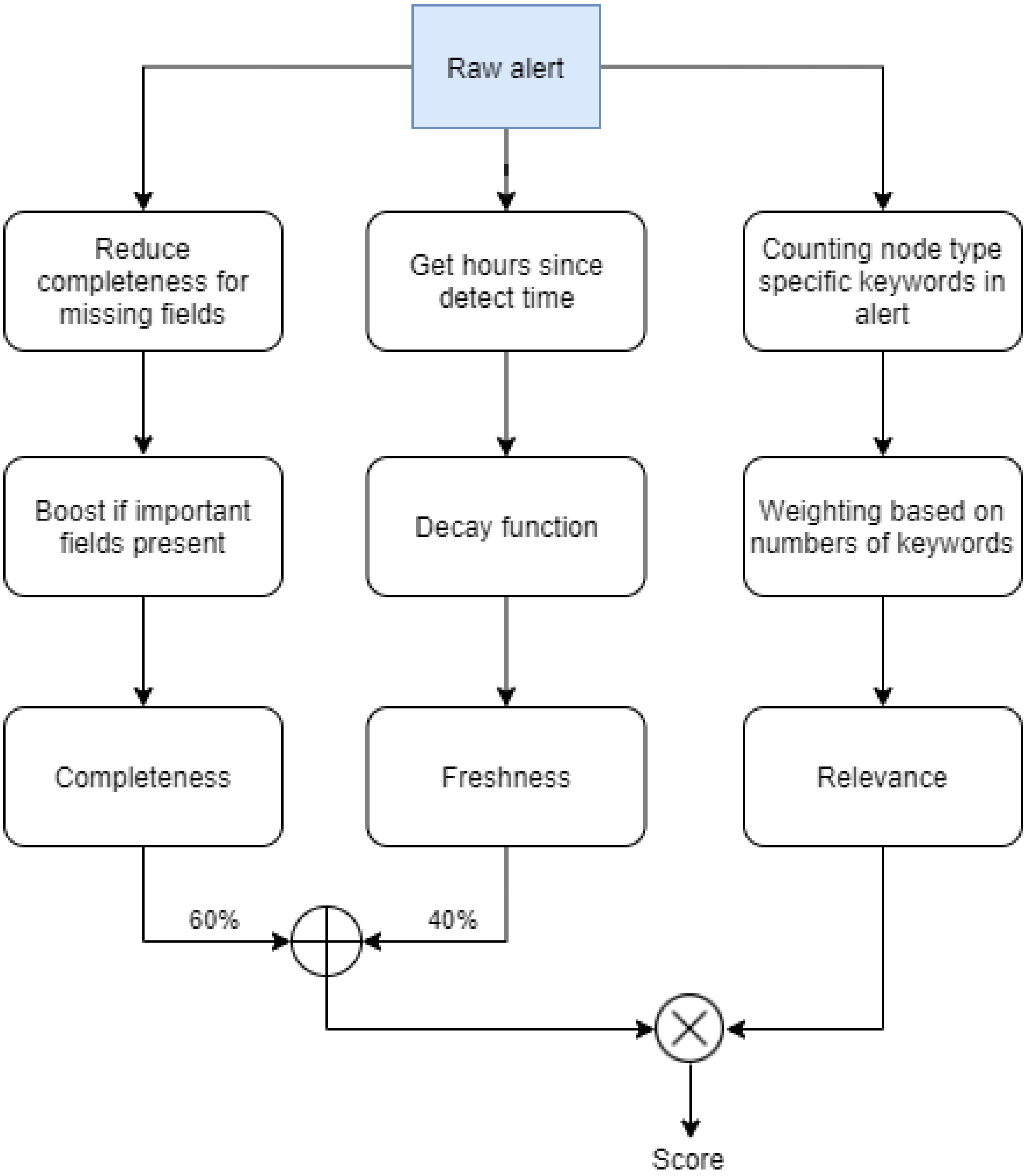

Raw alert
Reduce completeness for missing fields
Get hours since detect time
Counting node type specific keywords in alert
Boost if important fields present
Decay function
Weighting based on numbers of keywords
Completeness
Freshness
Relevance
60%
40%
Score

Figure 4.3: Alert Quality

In the Figure 4.3, Completeness, Freshness and Relevance are defined as:

- Completeness - measures how much the the TI complies with IDEA schema definition. Completeness decreases for each missing field.

- Freshness - measures how long the information is shared since it is generated. It decays as time passes.

- Relevance - for each type of alert source, a list of specific keywords is predefined along an associated relevance value. The relevance score is based on the number of keyword occurrences in the alert's field.

In this approach TIQ is expressed as

$$TIQScore = 60\% * Completeness + 40\% * Freshness) * Relevance \tag{4.1}$$

The quality score is calculated for every received security event. In the current, static community structure usage, events are exchanged continually between known community members and trust scores are used primarily as input to an alert prioritization scheme. In the decentralized scenario I anticipate blocks i.e. files of indicators of compromise being exchanged and the individual event quality scores being aggregated by the TIQ assessment function for such event files.

4.5 Reputation Assessment

TITAN uses the EigenTrust algorithm to calculate reputation score. EigenTrust takes in peer transaction histories and produces global trust values for all participants [51]. EigenTrust is based on the notion of transitive trust. Each peer calculates a local trust score value v_{ijk} i.e. the information quality value of k^{th} transaction from provider i to consumer j. The reputation that consumer j contributes to provider i is R_{ij}:

$$R_{ij} = \sum_k v_{ijk} \tag{4.2}$$

The transaction history of the files' quality scores forms a local reputation matrix R. EigenTrust computes global trust values using R, based on power iteration. The basic idea is formulated in following pseudocode.

Algorithm 1 Pseudocode of Eigen Trust

```
$\vec{t}^{(0)} \leftarrow \vec{e}$
repeat
    $\vec{t}^{(k+1)} \leftarrow R^T \vec{t}^{(k)}$
    $\theta \leftarrow ||\vec{t}^{(k+1)} - \vec{t}^{(k)}||$
until $\theta < \epsilon$
return $\vec{t}^{(k+1)}$
```

$\vec{t}^{(0)}$ represents the initial trust value of each peer. In each iteration, the original trust value is weighted by its friends' opinions. With large iterations, the trust vector $\vec{t}$ will converge to the same vector for every peer. Namely it will converge to the left principal eigenvector of R. $\vec{t}$ represents a global trust vector in this model [51].

TITAN stores the transaction trust scores on Ethereum. The integrity and transparency provided by the blockchain addresses the two main EigenTrust security concerns i.e. i) the current trust value of a peer must not be computed by and reside at the peer itself and ii) the calculation of a global trust score for any peer must not be miscalculated by any other peer [51]. A beneficial side effect of storing the local trust values in Ethereum is that the centralized version of the algorithm can be used to calculate the global trust score.

Every peer with the reputation algorithm TEE enclave is capable of processing Eigentrust and can store the result in Ethereum. A reward is provided by Ethereum to peers who calculate the Eigentrust for the community.

Chapter 5

TITAN Implementation

5.1 Introduction

This chapter explains the details of TITAN implementation. The following Figure 5.1 illustrates the overall structure.

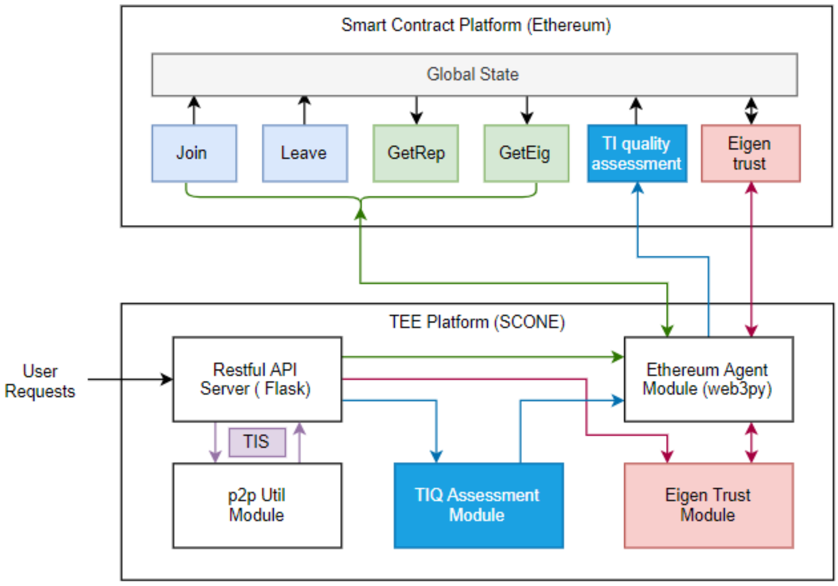


Figure 5.1: TITAN Implementation

TITAN consists of two components. The upper component is Ethereum part. A global state containing User list, TIQ and reputation scores is maintained by smart contract interfaces. The lower component is the TEE part, which contains five modules.

Further information about those components and modules will be detailed in subsequent sections. Here I will express the service flows. I have implemented a

Restful API Server to dispatch users' requests. There are mainly three different types of requests, labeled with three different colors:

- *Green Work Flow.* If the request is to interact with a smart contract directly, such as to fetch information, it will be assigned to Ethereum Agent Module, web3py, which will call the smart contract.

- *Blue Work Flow.* If a user wants to assess the alert quality, the request will be transmitted to TIQ Assessment Module. Afterwards, the result will be upload to Blockchain through web3py.

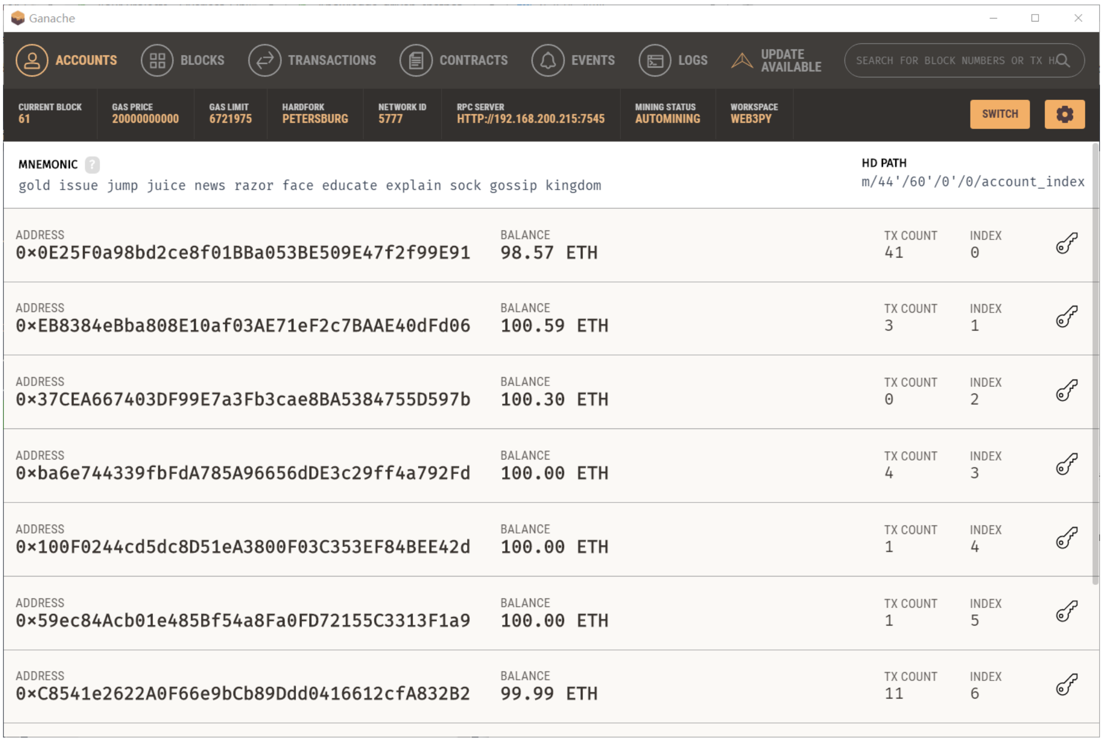
- *Red Work Flow.* When a user wants to update the reputation score, an EigenTrust Module is assigned to handle this request. It will fetch data from a smart contract, compute the Trust Score and upload them to Blockchain.

5.2 Ethereum Part

Ethereum is the most widely adopted platform to deploy Smart Contracts. I develop the TITAN Smart Contract using an Ethereum Blockchain. Algorithm 2 exhibits the pseudocode in appendix 1.

Since the transactions in the Ethereum main-net are costly and time-consuming, I don't deploy smart contracts in the main-net. The evaluation of Ethereum are given later in Chaper 6.

As a workaround, I deploy smart contracts in Ganache, an emulation Ethereum network [52]. Ganache will provide a personal blockchain for Ethereum development, where I can do anything as in Ethereum, such as send transactions, deploy and call smart contracts.



CURRENT BLOCK	GAS PRICE	GAS LIMIT	HARDFORK	NETWORK ID	RPC SERVER	MINING STATUS	WORKSPACE
61	20000000000	6721975	PETERSBURG	5777	HTTP://192.168.200.215:7545	AUTOMINING	WEB3PY

MNEMONIC
gold issue jump juice news razor face educate explain sock gossip kingdom

HD PATH
m/44'/60'/0'/0/account_index

ADDRESS	BALANCE	TX COUNT	INDEX
0×0E25F0a98bd2ce8f01BBa053BE509E47f2f99E91	98.57 ETH	41	0
0×EB8384eBba808E10af03AE71eF2c7BAAE40dFd06	100.59 ETH	3	1
0×37CEA667403DF99E7a3Fb3cae8BA5384755D597b	100.30 ETH	0	2
0×ba6e744339fbFdA785A96656dDE3c29ff4a792Fd	100.00 ETH	4	3
0×100F0244cd5dc8D51eA3800F03C353EF84BEE42d	100.00 ETH	1	4
0×59ec84Acb01e485Bf54a8Fa0FD72155C3313F1a9	100.00 ETH	1	5
0×C8541e2622A0F66e9bCb89Ddd0416612cfA832B2	99.99 ETH	11	6

Figure 5.2: Ganache Blockchain

The interaction between TITAN and smart contracts is executed by Ethereum Agent Module.

This Smart Contract can be divided into two interconnected parts: Global State and Interfaces.

Global State:

- userList - the list of joined users.
- leaveList - the list of left users.
- repList - the list of reputations for each joined user.
- eigList - the list of EigenTrust Score for each user.
- repMatrix - the aggregated reputations containing the provider - consumer relation.
- providerDataList - the data list provided by users.
- consummerDataList - the data list consumed by users.

- dataValue - the average quality score of each batch of data.

Interfaces:

- join - to register in TITAN system.
- leave - to leave the TITAN.
- getRep - to get a reputation score for a specific user.
- getEig - to get a EigenTrust Score for a user.
- updateRep - to upload a quality score of a particular batch of data.
- offloadRep - to offload the reputation matrix which is the input of EigenTrust module.
- updateEig - to store the results generated by EigenTrust module.

5.3 TEE Part

All the compute-intensive and data-intensive tasks are offloaded to the TEE component. The TEE part and Smart Contract Part communicate through Ethereum transactions.

All of those tasks are implemented in SCONE platform and regulated by a RESTful API Server.

5.3.1 SCONE

SCONE is a Secure CONtainer Environment for Docker using the Intel SGX technology proposed by Arnautov et al [53] in 2016. The innovation point of this platform is combining the features of container technology and TEE technology together. SCONE has three major desirable properties:

1. *A small TCB.* In SCONE, system calls are executed outside of the enclave, but they are shielded by transparently encrypting / decrypting application

data on a per-file-descriptor basis and network communication is protected by Transport Layer Security (TLS).

2. *A low overhead.* SCONE provides a user-level threading implementation which can maximize the time that threads spend inside the enclave and reduce costly enclave transitions of threads.

3. *Transparent to Docker.* Secure containers behave like regular containers in the Docker engine. SCONE requires only a SGX-capable Intel CPU, a SGX kernel driver and an optional kernel module for asynchronous call support.

To deploy TITAN on the platform of SCONE, instructions are presented as follows:

1. Install docker.

Official docker tutorial is the best practise. SCONE also provides simplified execution scripts for Ubuntu clients.

```
curl -fssl https://raw.githubusercontent.com/SconeDocs/SH/
   master/install_docker.sh | bash
```

2. Install Intel SGX driver

The official installation method of the SGX 2.0 driver on Linux distribution is given in its github page [54]. Alternatively, Ubuntu users could execute the following:

```
curl -fssl https://raw.githubusercontent.com/SconeDocs/SH/
   master/install_sgx_driver.sh | bash
```

To check if the SGX driver is successfully installed, following command is suggested.

```
ls /dev/isgx >/dev/null 2>1  && echo "SGX Driver installed" ||
     echo "SGX Driver NOT installed"
```

The CPU and the motherboard BIOS must support SGX, otherwise the installation will fail. In this case, *simulation mode* of SCONE allows prototype development.

3. Run a SCONE container

SCONE supports various programming languages including C, C++, Fortran, GO, Java, Python, PyPy, Node.js, R, Rust. For each language, SCONE provide different docker images, which is used to create Enclaves and running specific programs.

To run the Python interpreter inside an enclave in interactive mode, the container should be started.

```
docker run --rm -it -v "$PWD":/usr/src/myapp -w /usr/src/myapp
    sconecuratedimages/python:3.7-alpine sh
```

The application code located in *$PWD* is mapped into */usr/src/myapp* inside the container.

4. Execute inside an enclave

The commands will execute a python interpreter inside an enclave

```
SCONE_HEAP=256M SCONE_VERSION=1 SCONE_MODE=AUTO python
```

The environment variable is set as *SCONE_MODE=AUTO*, thus SCONE will use SGX enclaves when available and emulation mode otherwise.

The outputs indicate an enclave is successfully created.

```
export SCONE_QUEUES=4
export SCONE_SLOTS=256
export SCONE_SIGPIPE=0
export SCONE_MMAP32BIT=0
export SCONE_SSPINS=100
export SCONE_SSLEEP=4000
export SCONE_LOG=1
export SCONE_HEAP=268435456
export SCONE_STACK=2097152
export SCONE_CONFIG=/etc/sgx-musl.conf
export SCONE_ESPINS=10000
export SCONE_MODE=sim
export SCONE_ALLOW_DLOPEN=yes (unprotected)
export SCONE_MPROTECT=yes
musl version: 1.1.20
Revision: a1e5196e61af96370b5b380dc4a03e49542e4f6d (Tue Jul 23
    22:00:54 2019 +0200)
Branch: master (dirty)

Enclave hash:
   fa59564100f17f76284aeaa4fa335e10ebeb2888a5bb221354d3c757f8
   79b493
```

```
Python 3.7.5 (default, Oct 17 2019, 12:25:15)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more
    information.
>>>
```

5.3.2 RESTful API Server

REpresentational State Transfer (REST) is a software architectural style for building web services. REST is built on top of HTTP, i.e. the web services expose resources in the form of Uniform Resource Identifiers (URIs). The prominent advantage of REST architectural style is scalability which allows the support of large numbers of components and interactions among components.

TITAN adopts an internal RESTful API Server to integrate the sub-modules. The APIs is designed as follows:

Table 5.1: APIs of TITAN

Api	Methods	Description
/api/join	POST	To join in the platform.
/api/leave	POST	To leave the platform.
/api/updateRep	POST	To record the alert quality scores and maintain reputation matrix.
/api/updateEig	POST	To update the EigenTrust Values.
/api/getRep	GET	To get the peer's reputation score based on session.
/api/getEig	GET	To get the peer's EigenTrust value based on session.

Flask is one of the most popular lightweight Python web application frameworks, which are suitable to build the RESTful API Server.

To install Flask using pip:

```
pip install -U Flask
```

A quick example of Flask application is given as:

```
from flask import Flask, jsonify, request

app = Flask(__name__)
```

```
@app.route('/', methods=['POST'])
def hello_world():
    addr = request.form.get('addr')
    session['addr'] = addr
    return jsonify(
        addr=addr,
        info="Hello, This is Titan."
    )
```

5.3.3 Ethereum Agent Module

Web3.py is a python library for interacting with Ethereum. It is installed by pip:

```
pip install web3
```

The Smart Contract is written in Solidity language, then deployed through Web3.py. The process is detailed in Appendix 2.

During the deployment the *address* and *abi* of Smart Contract is stored, which are used to construct a contract instance during interacting with Ethereum. The contract instance is defined in Appendix 3.

5.3.4 P2P Util Module

P2P Util Module is designed to set up a P2P Network, through which Alert files are sharing. It acts as a basic Threat Intelligence Sharing Platform which is a substitute of a fully featured TIP.

To initiate a P2P network, various strategies are developed. The easiest way is to dedicate a central server, peers register through the server and get the information of other members. Then they communicate with each other directly. This is a hybrid P2P network. The second solution is to assign a specific port and send millions of initial searches in order to find other participants. It is a pure decentralised approach but with low availability and performance.

A list of seed peers needs to be predefined. Every newly instantiated node will send a message to one of the seeds, the seed will return a list of participants. After receiving the member list, the new comer will introduce himself to every one on the list. Figure 5.3 conveys the initiation process.

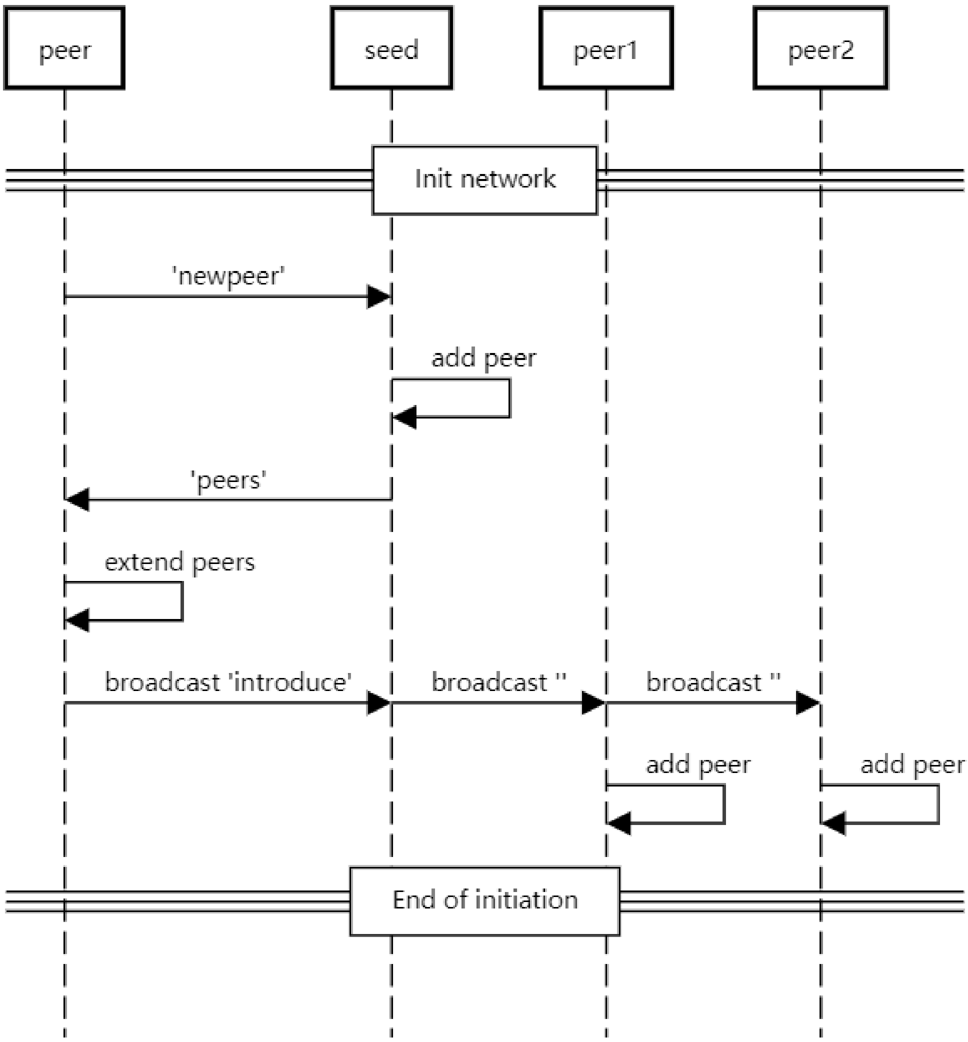


Figure 5.3: Initiate a P2P network

Peers need to react on various messages; thus, the messages are wrapped in a certain protocol: Each message contains two field. The first field indicates the command, and the second field contains the data. Those messages are transferred with json format.

```
{
    "type":"command string",
    "data":"content string",
}
```

Using this protocol, peers can chat and share file with each other, they can exit the network as well. The whole sequence diagram of each peer is given is Appendices 4.

5.3.5 TIQ Assessment Module

TIQ Assessment Quality is introduced to access the quality of alerts which are complied with Intrusion Detection Extensible Alert (IDEA) format. Horizon 2020 (H2020) PROTECTIVE project developed a security situational awareness manager to defend cyber attacks [47]. This open source project contains a PROTECTIVE-trustmodule which addressed the alert quality issue [55]. I adopt this module into TITAN. The output of this module will be stored in the Ethereum part. To reduce the transaction traffic of TITAN, alerts are shared and assessed in bundles of thousands. This perimeter can be defined by user. The average score of a bundle of alerts will be sent to Ethereum.

The below function is to split a big file into bundles:

```
# given the folder name, file name, lines (default = 1000),
   return a list of small file.
def split(path, filename, lines = 1000):
    smallfile = None
    with open(path + filename) as bigfile:
        for lineno, line in enumerate(bigfile):
            if lineno % lines == 0:
                if smallfile:
                    smallfile.close()
                small_filename = path + ’small_file_{}.txt’.
   format(lineno + lines)
                smallfile = open(small_filename,’w’)
            smallfile.write(line)
        if smallfile:
            smallfile.close()
```

After that, the average score of a bundle file will be assessed.

```
# given a bundle of alerts, return its average score.
from protective_trustmodule.alertquality import AlertQuality,
   AlertQualityConfig
def assess(filename):
    AQ = AlertQuality(AlertQualityConfig())
    allscore = []
    with open(filename, ’r’) as f:
        for i in f:
            allscore.append( AQ.get_quality(i) )
    sum = 0
    for j in allscore:
        print(j[’Score’])
        sum += j[’Score’]
    return sum/len(allscore)
```

5.3.6 EigenTrust Module

EigenTrust is a reputation calculation algorithm developed by Karmvar et al. [51]. Due to the feature of preventing collusion attack that EigenTrust has, it is chosen as TITAN's reputation assessment algorithm.

The concept and algorithm of EigenTrust is introduced in section 4.5.

Because Smart Contracts in Ethereum can not return 2-dimensional list currently, EigenTrust Module fetches one-dimensional list as input. The list is converted to Matrix through:

```
import numpy as np

def list2Mat(list):
    l = len(list)
    lMat = int(np.sqrt(l))
    Mat = np.reshape(list,(lMat,lMat))
    return Mat
```

The matrix must be normalized before procession:

```
def normalizeMartix(repMat):
    resMat = []
    ## the sum of every row is equal to 1.
    for row in repMat:
        s = sum(row)
        if s != 0:
            row = [x/s for x in row]
        # if i doen't trust anyone, i trust them equally.
        else:
            row = [1.0/len(row)]*len(row)
        resMat.append(row)
    return resMat
```

Then EigenTrust Module produces the trust value for all peers through:

```
def eigenTrust(normMat, epsilon=1e-8):
    npMat = np.array(normMat)
    transMat = npMat.T
    l = len(normMat)
    t0 = [1/l] * l
    tk = t0
    tk1 = None
    delta = 1
    while delta > epsilon:
        tk1 = transMat.dot(tk)
        delta = np.linalg.norm(tk1 - tk)
        tk = tk1
    return tk
```

Chapter 6

Evaluation

TITAN provides trust services to a TI sharing platform. The H2020 PROTECTIVE project is used as the TI sharing layer to validate the approach. The introduction of PROTECTIVE and the cooperation of PROTECTIVE and TITAN is explained in section 6.1.

TITAN is an innovative trust management system combining Smart Contract and Trust Computing. Few applications are developed using such an approach. This situation leads to hardly any horizontal comparisons. Nonetheless I test the performance of TIQ function of phase B (see Figure 4.2) in section 6.2, and demonstrate other functionalities of TITAN there.

All the results of TITAN are stored in Blockchain through transactions. The performance of phase C (see Figure 4.2) in TITAN highly relies on Blockchain Performance. Since the call of Smart Contracts takes a lot of time to be mined, usually more then 3 seconds, the Smart Contract is the performance bottleneck of phase C. Thus, the evaluation of Ethereum is required. The throughput and transaction latency of Ethereum are assessed in section 6.3.

6.1 PROTECTIVE

PROTECTIVE is a Security Situational Awareness Manager (SSAM) designed to provide security teams with a greater cyber defence capability through improved cyber situational awareness (CSA) in order to raise their level of awareness of the risk to their business posed by cyber-attacks as well as enhancing their capacity to respond to threats.

PROTECTIVE has been developed to support two different modes of deployment, Peer-2-Peer (P2P) and Centralized Mode. TITAN is compatible with both modes. Only the P2P architecture is introduced here. The Peer-to-peer architecture represents separate communities, each with their own PROTECTIVE Node – see figure 6.1 below.

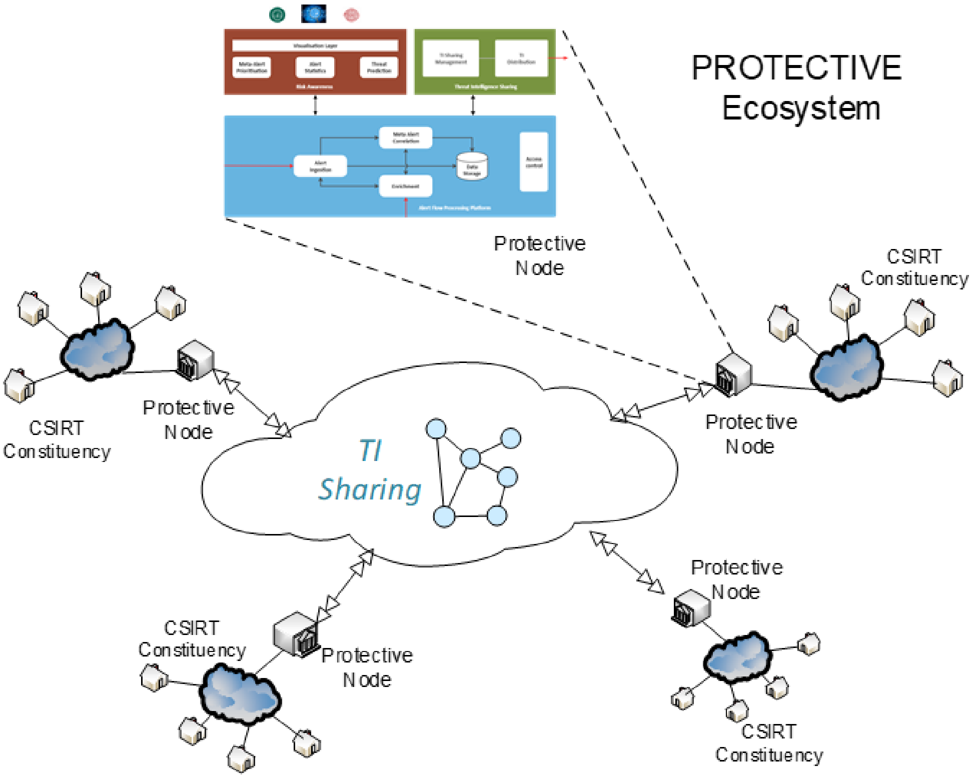


Figure 6.1: PROTECTIVE Ecosystem

Such an ecosystem is a federation of a number of PROTECTIVE nodes in different partner organisations which share information for the purposes of mutually improving identification and prevention and mitigation of threat events in their respective constituencies. Each network has at least one PROTECTIVE node which is used to raise organisational cyber situational awareness and also to route threat information to and from community partner PROTECTIVE node. This is depicted by the double ended arrows. The figure 6.2 shows the details inside an instance of a PROTECTIVE node.

The PROTECTIVE node is designed as an information processing pipeline.

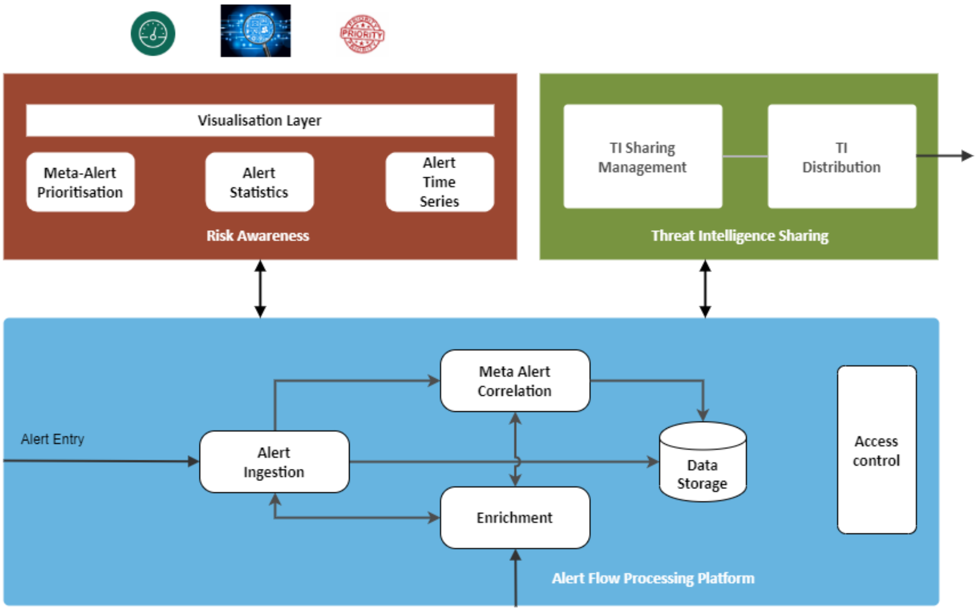


Figure 6.2: PROTECTIVE Node

Information enters the Alert Flow Processing system at the bottom left side of the diagram through the Alert Ingestion subsystem. The incoming alerts are converted to the PROTECTIVE normalised security alert format, IDEA (Intrusion Detection Extensible Alert). After ingestion, IDEA alerts are passed to the Enrichment subsystem. Here, additional data is annotated to the alerts to aid with their further processing. After enrichment the alerts are passed to the Meta Alert Correlation for further processing. In this module IDEA alerts are aggregated into composite structures known as Meta-Alerts.

The alerts and Meta-Alerts are used by the analysis features described above. Security alerts are routed to the TI Sharing subsystem for distribution.

6.1.1 Interaction between PROTECTIVE and TITAN

As described in Chapter 4, there are three layers in TITAN architecture. The first layer is TI sharing layer which is in fact outside of its framework and instead consumes the TITAN trust services. The PROTECTIVE ecosystem works in this layer.

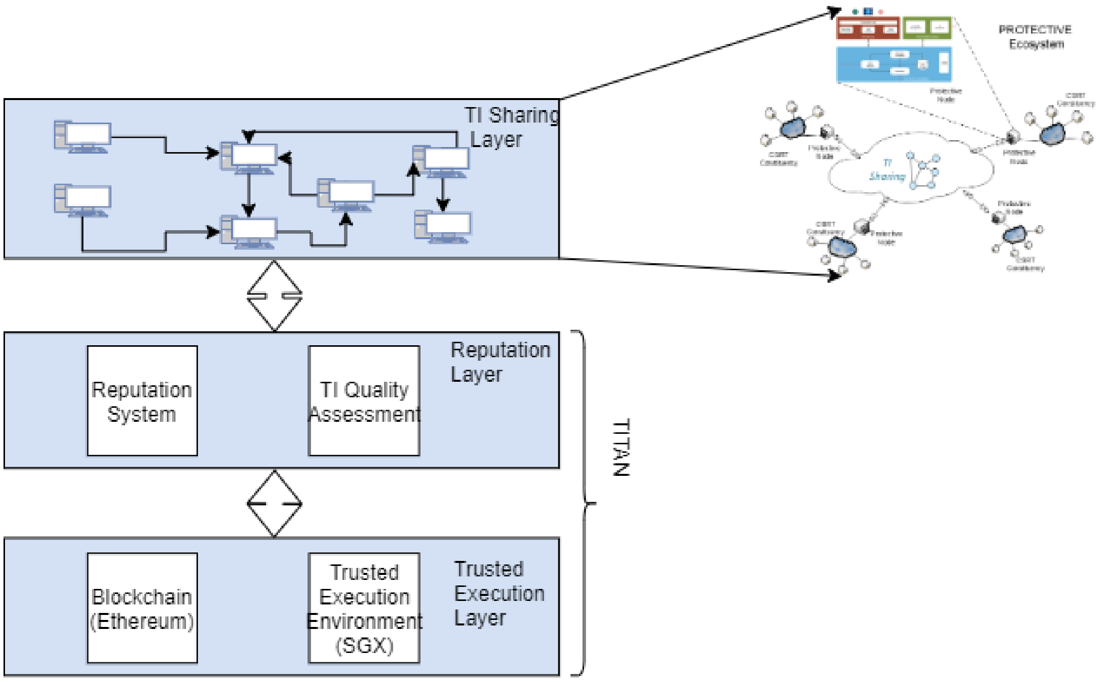


Figure 6.3: TITAN and PROTECTIVE

When the threat intelligence is shared from one PROTECTIVE node to another, TITAN will access the quality of alerts and calculate the reputation for each entity.

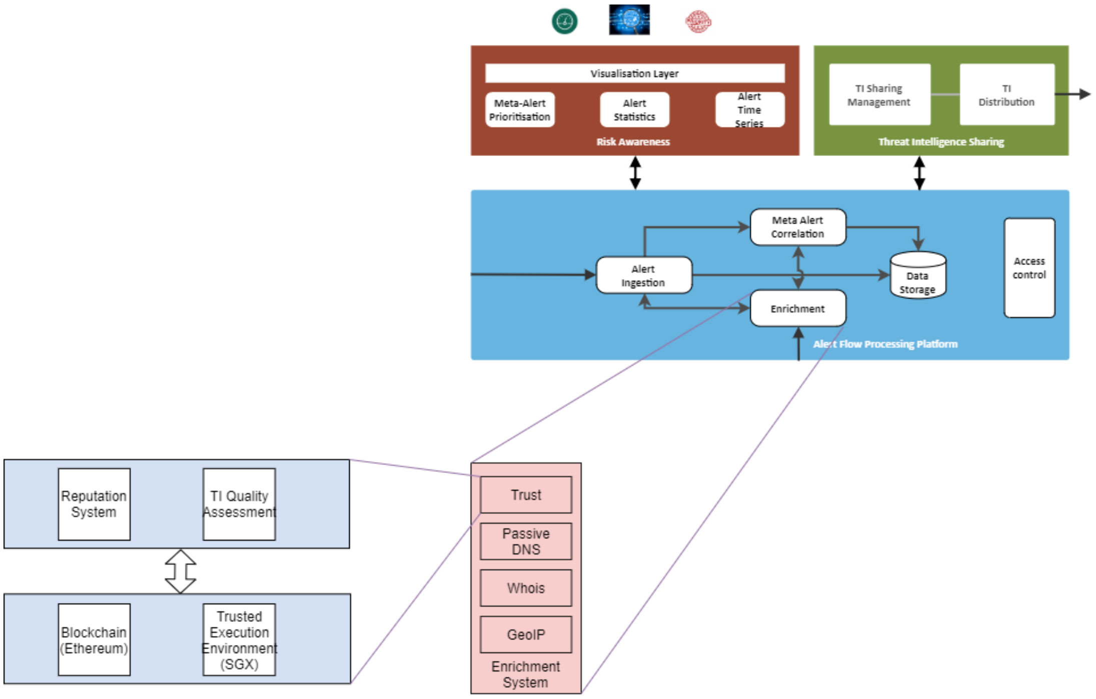


Figure 6.4: Connection between PROTECTIVE and TITAN

Figure 6.4 shows the connection between PROTECTIVE and TITAN. When the IDEA alerts flow through the PROTECTIVE enrichment system, they will be dispatched to TITAN. TITAN provides trust services for the PROTECTIVE enrichment system, especially the trust sub-module.

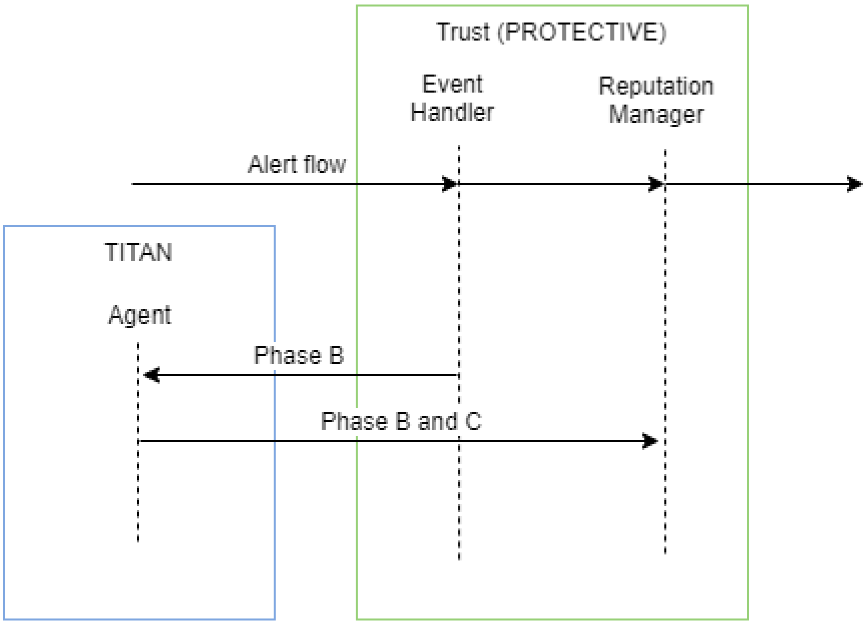


Figure 6.5: Alert flow between PROTECTIVE and TITAN

When the alerts flow through trust sub-module, the event handler dispatches a copy of alerts to TITAN, as show as Figure 6.5. TITAN will assess the alerts and then send back TIQ and EigenTrust scores to the Reputation Manager. Phase B and C are the illustrated in Figure 6.6 below. They are very much similar to the phases in Figure 4.2 which is detailed in section 4.3.

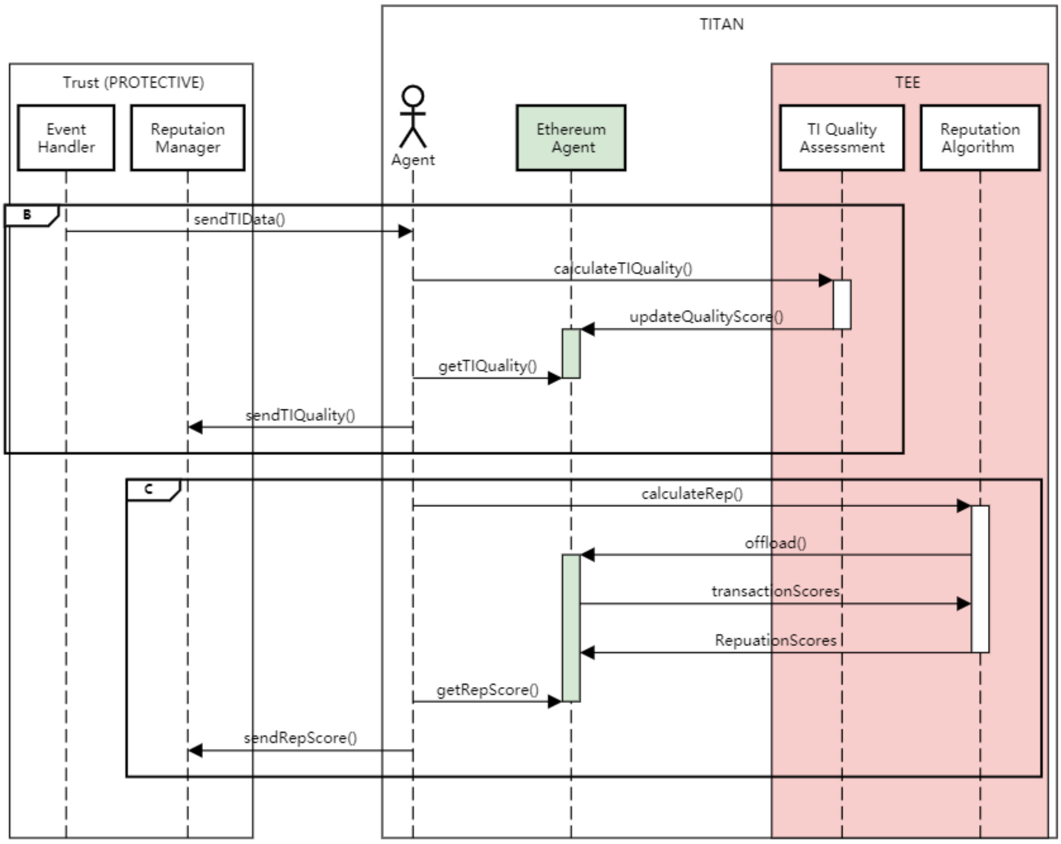


Figure 6.6: Details of Alert flow between PROTECTIVE and TITAN

In phase B, the Event Handler of PROTECTIVE sends Batches of alerts to the Agent of TTIAN, when the agent gets the result, it will send the TIQ back to the Reputation Manager. At some later time, in phase C, after the agent of TITAN calculates the latest EigenTrust scores, it will send these scores to the Reputation Manager.

How the TIQ and the EigenTrust are utilized depend on the design of Reputation Manager. The Information Exchange Policy, which governs all the exchange of data as previously described in section 3.4.3, can work based on EigenTrust. For example, a discreet peer will only receive files from a node whose EigenTrust is above average score. The ranking of alerts may take the TIQ and EigenTrust as a valuable reference. Besides that, a micro-payment system may work based on TIQ. This system can be built on (but not limited to) the Smart Contract. There are various use-case scenarios which are beyond the scope of TITAN.

6.2 TITAN Verification

The dataset used to evaluate TITAN is the source Threat Intelligence file shared in H2020 PROTECTIVE project. It is a cluster of alerts using the IDEA format. Normally a single file is as large as several GBs with around 200,000 alerts. A single alert is a JSON serialized IDEA security event, as in this example:

```
{
   "Format": "IDEA0",
   "ID": "4390fc3f-c753-4a3e-bc83-1b44f24baf75",
   "CreateTime": "2012-11-03T10:00:02Z",
   "DetectTime": "2012-11-03T10:00:07Z",
   "WinStartTime": "2012-11-03T05:00:00Z",
   "WinEndTime": "2012-11-03T10:00:00Z",
   "EventTime": "2012-11-03T07:36:00Z",
   "CeaseTime": "2012-11-03T09:55:22Z",
   "Category": ["Fraud.Phishing"],
   "Ref": ["cve:CVE-1234-5678"],
   "Confidence": 1,
   "Note": "Synthetic example",
   "ConnCount": 20,
   "Source": [
      {
         "Type": ["Phishing"],
         "IP4": ["192.168.0.2-192.168.0.5", "192.168.0.10/25"]
   ↪  ,
         "IP6": ["2001:0db8:0000:0000:0000:ff00:0042::/112"],
         "Hostname": ["example.com"],
         "URL": ["http://example.com/cgi-bin/killemall"],
         "Proto": ["tcp", "http"],
         "AttachHand": ["att1"],
         "Netname": ["ripe:IANA-CBLK-RESERVED1"]
      }
   ],
   "Target": [
      {
         "Type": ["Backscatter", "OriginSpam"],
         "Email": ["innocent@example.com"],
         "Spoofed": true
      },
      {
         "IP4": ["10.2.2.0/24"],
         "Anonymised": true
      }
   ],
   "Attach": [
      {
         "Handle": "att1",
         "FileName": ["killemall"],
         "Type": ["Malware"],
         "ContentType": "application/octet-stream",
         "Hash": ["sha1:0c4a38c3569f0cc632e74f4c"],
         "Size": 46,
         "Ref": ["Trojan-Spy:W32/FinSpy.A"],
         "ContentEncoding": "base64",
         "Content": "TVpqdXN0a21kZGluZwo="
      }
   ],
   "Node": [
      {
         "Name": "cz.cesnet.kippo-honey",
         "Type": ["Protocol", "Honeypot"],
         "SW": ["Kippo"],
         "AggrWin": "00:05:00"
      }
   ]
}
```

Such dataset is divided into small files containing thousands of alerts. Those granulated files will be assessed through TITAN in batches.

I don't have a PROTECTIVE system running so that I emulate it by running a *P2P Util Module* which I developed by myself. The *P2P Util Module* is fine to use because I am using the real PROTECTIVE events and the conclusion can be generalised. It works as a primitive threat intelligence sharing network. when a file was sent from a provider to a receiver. The TIQ assessment module takes in the file and assesses its quality. The whole process is depicted as a sequence diagram in appendix 4. When the quality value is generated it will send to the Ethereum blockchain. Afterwards, EigenTrust values will be calculated based on those transactions. The process works as shown in Figure 6.7.

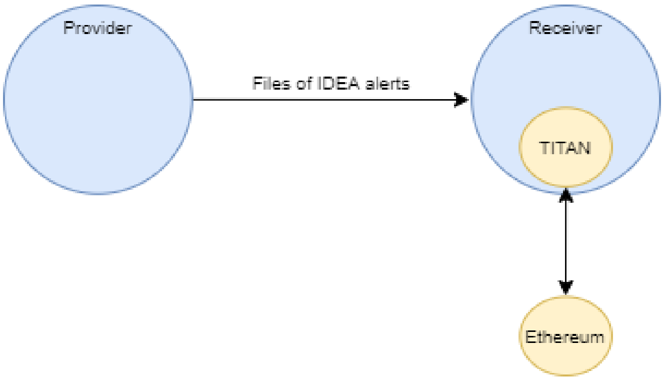


Figure 6.7: The Demo of TITAN

I took a file containing 100,000 alerts to test the performance of the TIQ algorithm which is the core program in phase B. The overall runtime is 163.743s, which means it takes 1.637 millisecond per alert. This result is based on my laptop, as specified in Table 6.1. Since a server may process higher-speed CPUs and larger memory, the runtime can be remarkably diminished.

Table 6.1: Computer Information

Items	Value
Computer Model	Timi TM1701
CPU	Intel(R) Core(TM) i5-8250U
Clock Speed	1.60 GHz
Core Number	4
RAM	8 Gbytes

In order to reduce the number of Blockchain transactions, the alerts are bundled into thousands. For the purpose of analyzing the effect that the bundling brings to the overall performance, I split the same file with different sizes, and record the length time that TIQ modules takes when assessing them. The result is depicted in Figure 6.8:

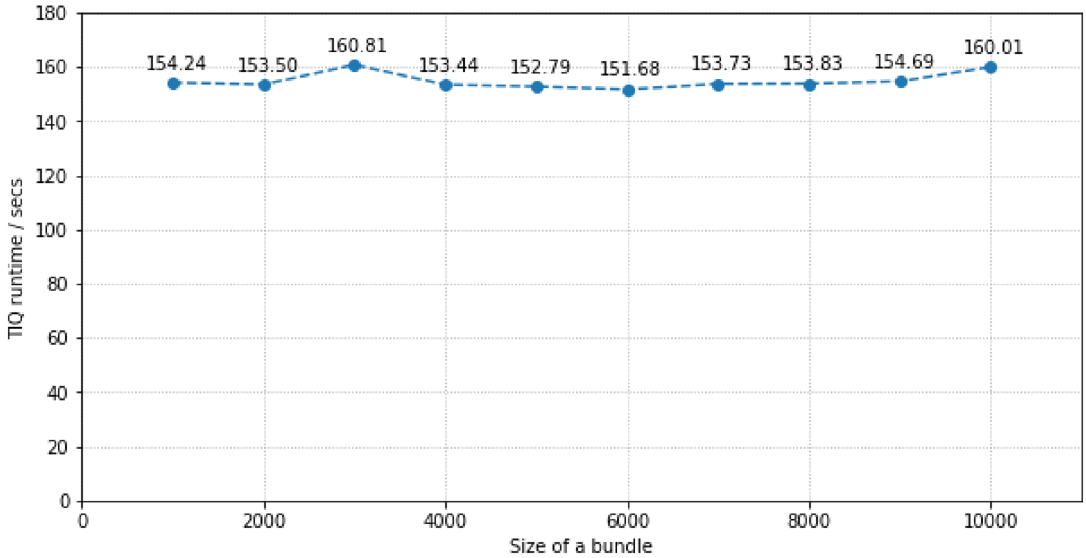


Figure 6.8: TIQ Runtime

The overall runtime of assessing a fixed number of alerts remains stable while the size of a bundle growing, which indicates that there is marginal effect that size of a bundle brings to the TIQ performance. The TIQ overall runtime is proportional to the volume of alert. The batch size is configurable with the possibility to turn it off when needed. For some security system, if the coming alerts are more massive and less sensitive, the batch size can be set larger. Otherwise, the batch size can be smaller. If the computing power are stronger, the batch size can be larger also.

Besides TIQ algorithms, other services of TITAN are highly interconnected with Ethereum. They are organized as RESTful APIs, which are demonstrated below.

Join in TITAN

In order to maintain one's reputation and trust score through TITAN, registration is required for every user. Since registration will call the smart contract causing transaction fees, a payable Ethereum address is indispensable.

```
POST /api/join
```

Table 6.2: /api/join Parameters

Name	Type	Description
addr	string	Required. It should be a payable Etheruem address possessed by the user.

Example:

```
{
    "addr":"0x0E25F0a98bd2ce8f01BBa053BE509E47f2f99E91"
}
```

Response:

```
Status: 200 OK

{
  "addr": "0x0E25F0a98bd2ce8f01BBa053BE509E47f2f99E91",
  "info": "You have successfully joined the network."
}
```

If the address already exists in our system, the user will be informed with its index.

```
Status: 200 OK

{
  "addr": "0x0E25F0a98bd2ce8f01BBa053BE509E47f2f99E91",
  "index":4,
  "info": "You had registered in the network."
}
```

After this function, a session containing use's address is created and ready for later use.

Leave TITAN

"Leave" API is called when one wants to conceal all of one's information on TITAN. Because blockchain records the history of transactions, it is unable to erase one's trace. When one leaves TITAN, its reputation and trust score is inaccessible through TITAN services.

```
POST /api/leave
```

No parameters are required when invoking this API. the address is obtained through the session mention above.

Response:

```
Status: 200 OK

{
  "addr": "0x0E25F0a98bd2ce8f01BBa053BE509E47f2f99E91",
  "info": "You have successfully left the network."
}
```

Upload TIQ

After receiving a TI file from a provider, the recipient may access the Alert quality for this file, and the result will send to Smart Contract through this function.

```
POST /api/updateRep
```

Table 6.3: /api/updateRep Parameters

Name	Type	Description
dataId	string	Required. It is the hash signature of a shared file.
dataValue	float	Required. This value is in [0,1). it is generated by TIQ assessment module.
uid1	string	Required. It is the id of the information provider which has already been registered.
uid2	string	Required. It is the id of the information consumer which has already been registered.

Example:

```
{
    "dataId":"e01a69a18e2600f1b3c155014bc3432109dcf3de9ab60616
   ↪  18258ff39cabb720",
    "dataValue":"0.860609007071295 8",
    "uid1":"0xEB8384eBba808E10af03AE71eF2c7BAAE40dFd06",
    "uid2":"0x37CEA667403DF99E7a3Fb3cae8BA5384755D597b"
}
```

Response:

```
Status: 200 OK

{
    "addr": "0x0E25F0a98bd2ce8f01BBa053BE509E47f2f99E91",
    "info": "The data quality value has been recorded."
}
```

The sum of TIQs of a certain provider is his reputation. The provider's reputation is aggregated when this API is successfully invoked. the reputation list of all participants will be used to calculate EigenTrust for each other.

Update EigenTrust

When a user request to update EigenTrust, the reputation scores hosted in blockchain will be fetch to TEE. Afterwards EigenTrust Module will computer the EigenTrust score for everyone. The latest scores will be upload to blockchain and replace the older version.

```
POST /api/updateEig
```

No parameters are needed.

Response:

```
Status: 200 OK

{
  "addr": "0xEB8384eBba808E10af03AE71eF2c7BAAE40dFd06",
  "info": "The EigenTrust Values have been updated."
}
```

Get Reputation

TITAN providers an interface for a user to fetch his reputation scores.

```
GET /api/getRep
```

Response:

```
Status: 200 OK

{
  "addr": "0xEB8384eBba808E10af03AE71eF2c7BAAE40dFd06",
  "score": 13
}
```

Get EigenTrust Value

After “updateEig” API is called, the EigenTrust Value is updated. It is fetched by:

```
GET /api/getRep
```

Response:

```
Status: 200 OK

{
  "addr": "0xEB8384eBba808E10af03AE71eF2c7BAAE40dFd06",
  "aveScore": 0.2,
  "value": 0.32499999
}
```

The sum of all member's EigenTrust value is equal to 1. “aveScore”, the average EigenTrust value, is given as a benchmark reference.

6.3 Ethereum Evaluation

The performance of phase C highly relies on Blockchain Performance. Since the call of Smart Contracts takes a lot of time to be mined, usually more then 3 seconds, the Smart Contract is the performance bottleneck of phase C. Thus, the evaluation of Ethereum is required. The throughput and transaction latency of Ethereum are assessed in this section.

In order to address the decentralized consensus problem, various consensus algorithms are deployed in Blockchain implementations. Peers need to compete for the right of composing blocks and validate transactions, which results in the limitation of the theoretical maximum number of Transactions Per Second (TPS). Bach et al. surveyed ten top high-profile Blockchain implementations and gave a comparison table in 2018 [56].

Table 6.4: TPS of several cryptocurrencies

Cryptocurrency Name	Protocol	TPS
Bitcoin	PoW	7
Ethereum	PoW	15
Ripple	RPCA	1500
Bitcoin Cash	PoW	60
Cardano	PoS	7
Stellar	SCP	1000
NEO	DBFT	10000
Litecoin	PoW	56
EOS	DPoS	∼millions
NEM	PoI	4000

Although the ideal TPS of Ethereum is 15, the practical TPS given by etherscan.io is 8.8 [57] and TPS rendered by etherchain.org is 6.9 [58] at the time of writing. The high the TPS, the shorter time a transaction validation takes.

In Ethereum, the waiting time of a transaction is affected by the Gas Price. Theoretically, the higher a transaction's Gas Price is, the more likely it will be included in the block that the miners are mining, i.e. the less waiting time it takes.

Etherscan provides a pending transactions pool which contains the details of more than 25000 pending transactions [59]. Fetching All those data gives a snapshot of pending transactions' status.

The following table illustrates the contents of this database:

Table 6.5: Pending Transactions

Index	Txn Hash	Nonce	Gas Limit	From	To	Datetime	Waiting Time /secs	Gas Price /Gwei	Value /Ether
0	0x548d...	131	52445	0xf256...	0x9a02...	2020-02-27 ...	1	8	0
1	0xbd26...	217	55478	0xea13...	0xada6...	2020-02-27 ...	1	3.96	0
2	0x90b2...	2	40000	0x7a83...	0x7788...	2020-02-27 ...	1	9	0
3	0x5128...	21	21000	0x98cf...	0xe830...	2020-02-27 ...	1	10	0.2701
4	0xf46c...	2189	187772	0xcc28...	0xe5ad...	2020-02-27 ...	1	8	0
...	...	...	...	...	...	...	...	...	...
25095	0x6606...	13	38948	0xff5b...	0xa689...	2020-02-20 ...	633600	0.054906	0
25096	0x7385...	3	100000	0xc55a...	0x71ae...	2020-02-20 ...	633600	0.01	0.000231
25097	0x9fba...	4	200000	0xef36...	0xac9b...	2020-02-20 ...	633600	0.01	0
25098	0x0eac...	7	200000	0xcc24...	0xac9b...	2020-02-20 ...	633600	0.01	0
25099	0x6b39...	3	21000	0xeb29...	0xee56...	2020-02-20 ...	633600	0.05	0.000273

the statistical information from the database is concluded as:

Table 6.6: Statistical Information of All Pending Transactions

	Gas Limit	Waiting Time / secs	**Gas Price / Gwei**	Value / Ether
count	22170	22170	22170	22170
mean	470659.3961	36175.20573	0.433291	0.297001
std	1407233.05	62528.17409	5.30478	3.084242
min	21000	1	0	0
25%	100000	28920	0	0
50%	136272	32040	0.01	0
75%	200000	32400	**0.1**	0.06
max	10000000	633600	328.008578	365

Conclusion 1: There are 75% transactions whose Gas Prices are below 0.10 Gwei. Their chance of being recorded on chain is almost impossible, thus they

can be deemed as insincere transaction. There are generated by accident or for experiment purpose and etc.

To evaluate the real transaction process, I select the data where $GasPrice > 0.1 Gwei$ as sincere transactions. The number of transaction drops from 22170 to 4985 (22.48%).

Table 6.7: Statistical Information of Sincere Pending Transactions

	Gas Limit	Waiting Time / secs	**Gas Price / Gwei**	**Value / Ether**
count	4985	4985	4985	4985
mean	737905.9739	**45063.07543**	**1.877072**	0.131827
std	1923154.778	78155.72702	11.067001	3.519321
min	21000	1	0.1	0
25%	60000	6480	0.3	0
50%	140000	30480	0.699467	0
75%	404540	32340	**1**	**0**
max	10000000	633600	328.008578	200.02772

Conclusion 2: The average Gas Price of all sincere pending transaction is 1.87 Gwei, and the waiting time is 45063s (12 hours 31 minutes 1 second).

Conclusion 3: More than 75% of transactions' value is 0.000000 Ether. It means the majority of transactions are micro-payment or non-payment transactions.

Most of the Ethereum users have an expectation that their transactions be validated as soon as possible. Therefore, the time frame of first 60 seconds is more valuable then the rest.

After select and analyse transactions in this time frame, I find:

Conclusion 4: For the pending transactions within the first 60 seconds, the average Gas Price is 8.83 Gwei, and the waiting time is 20.96s.

To identify the correlation between other factors with Waiting Time, I compute Pearson Correlation Coefficient, which is a general measure of the linear correlation between two variables.

Table 6.8: Pairwise Pearson Correlation Coefficient

	Gas Limit	**Gas Price / Gwei**	Value / Ether
Waiting Time / secs	0.014347	**-0.406881**	-0.054424

Conclusion 5: The correlation coefficient between "Gas Price" and "Waiting time" is -0.41, which indicates there is a medium negative correlation. It proves that the less the Gas Price is, the longer the transaction waits.

Conclusion 6: The effect that "Gas Limit" and "Value" bring to "Waiting time" is insignificant, with correlation coefficient of 0.014, -0.054 respectively.

To visualize the relation between "Gas Price" and "Waiting time", I draw the scatter figure as below.

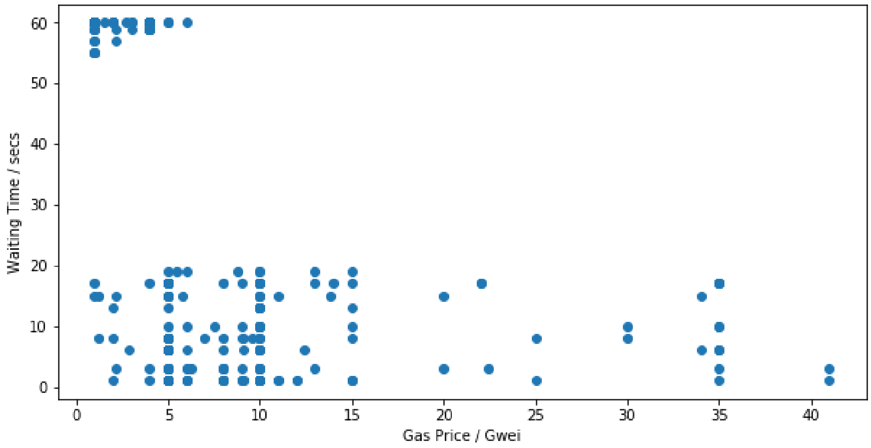


Figure 6.9: The relation between Gas Price and Waiting Time

Conclusion 7: From Figure 6.9, I conclude that when the Gas Price is above 7 Gwei, the waiting time is high likely blow 20 seconds. On the contrary, if the Gas Price is below 7 Gwei, there is a chance of waiting longer than 50 seconds.

The frequency histogram of Gas Price is given below:

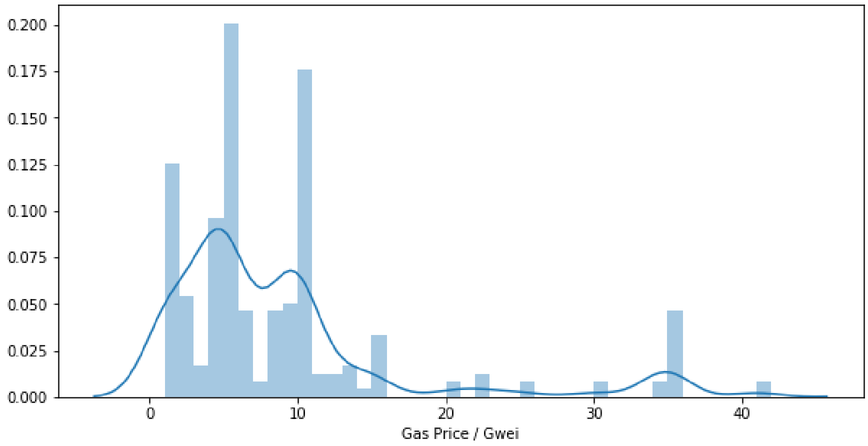


Figure 6.10: The frequency histogram of Waiting Time

Conclusion 8: In the first 60s-time frame, the Gas Price of highest probability is around 5, or 6 Gwei.

After aggregating the Gas Price according to waiting time, I compute the average Gas Price of each time interval.

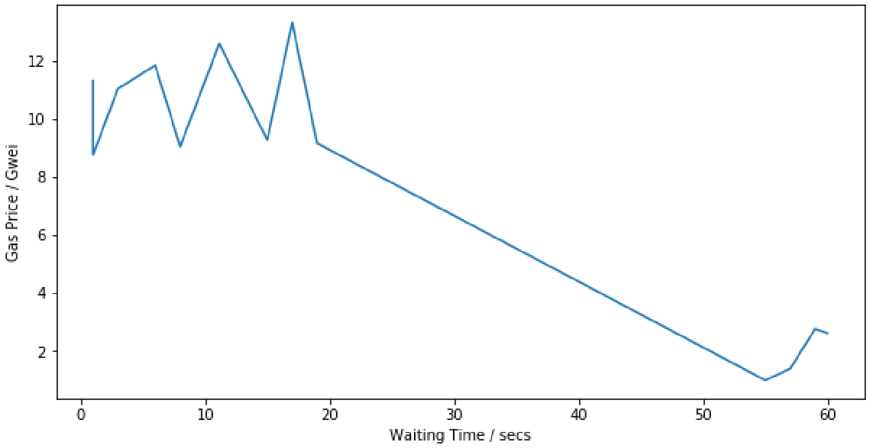


Figure 6.11: The average Gas Price of time intervals

Conclusion 8: From above figure, it is suggested that the waiting time decreases while the Gas Price rises. But when Gas Price is higher than 9 Gwei, there is no obvious decrease of waiting time as long as the Gas Price grows.

6.4 Conclusion

In this chapter, I use PROTECTIVE to verify the feasibility of TITAN solution. When the IDEA alerts flow through the PROTECTIVE enrichment system, they will be dispatched to TITAN. TITAN assesses the alerts and then send back TIQ and EigenTrust scores to the Reputation Manager of PROTECTIVE. The Reputation Manager will make full use of the trust services. For example, the Information Exchange Policy can work based on EigenTrust. A micro-payment system may be established based on TIQ.

Afterwards, I evaluate the performance of TIQ algorithm which is most critical and most frequently used in TITAN. It takes 1.637 millisecond to assess an event. The experiment proves that the TIQ performance is rarely affected by the batch size. So that the batch size is configurable regardless of the TIQ performance, but the characteristics of alerts as well as computing power of the system etc. If the alerts are less sensitive or the computing power are more strong, the batch size should be larger.

TITAN is system built on top of Ethereum, thus the throughput and transaction latency of Ethereum are also evaluated. The waiting time decreases while the Gas Price rises. But when the Gas Price is higher than 9 Gwei, there is no obvious decease of waiting time as long as the Gas Price grows.

The batch size is higher, the number of transactions is fewer. In order to reduce the time and cost caused by Blockchain transactions, it is suggested that the batch size should be larger when it is available. In order to reduce unnecessary transactions, the call of *Update EigenTrust* is better to be restricted by a certain time frame.

Chapter 7

Conclusion and Discussion

In this final chapter, I firstly recap the whole research project in section 7.1. Then I discuss some findings, some limitations of TITAN and few expectations of future researches in section 7.2.

7.1 Conclusion

Threat Intelligence Sharing is one of the most promising approaches to resist ever-growing cyber attacks. Many pressing challenges still exist for effective TI sharing especially around trust. First of all, I identify three types of trust requirements for a general TIP, which are *Trust between Participant*, *Trust in the TI Platform* and *Trust of TI Quality*. After studying various blockchain and trust computing implementations, I design TITAN to satisfy each demand.

TITAN is an innovative decentralized TI sharing trust enhancement framework combining Blockchain and Trust Computing, which consists of three layers: *TI Sharing Layer*, *Reputation Layer* and *Trusted Execution Layer*. The Blockchain stores the trust and reputation scores which is tamper-proof. The TEE securely hosts TI quality scoring algorithm and the reputation scoring algorithm.

I implemented TITAN with Ethereum Blockchain and Intel SGX. TIQ algorithm and EigenTrust algorithm are adopted as the reputation algorithms. The TITAN Smart Contract is also developed to store the TIQ and EigenTrust scores.

In the evaluation phase, the feasibility of TITAN solution is verified by PRO-

TECTIVE. The interaction of PROTECTIVE and TITAN is illustrated and explained. Some typical use-cases of TITAN are also introduced.

Afterwards, I evaluated the performance of TIQ algorithm which is most critical to TITAN. The experiment proves the TIQ performance is rarely affected by the batch size. So that the batch size is configurable regardless of the TIQ performance, but the characteristics of alerts as well as computing power of the system etc.

The throughput and transaction latency of Ethereum are also evaluated. It is proven that the waiting time decreases as the Gas price rises. But it won't drop too much when the Gas price is higher than 9 Gwei. In order to reduce the time and cost caused by Blockchain transactions, it is suggested that the batch size should be larger when it is available. And the call of *Update EigenTrust* is better to be restricted by a certain time frame.

7.2 Discussion

There are also some findings worth discussing during this research:

1, The combination of Smart Contract and TEE can be widely used.

Smart Contract has two main limitations:

- It can not access off-chain data.
- Storage and computation on smart contracts are expensive.

Therefore, Smart Contract alone is not suitable for data-intensive or compute-intensive jobs. I introduce Trusted Execution Environment to off-load those jobs from smart contracts. TEE is a hardware isolated execution environment. It only needs Minimal Trusted Computing Base which is Processor. And It provides a programmable platform, the application's integrity and confidentiality is maintained.

2, Currently TITAN is based on Intel SGX.

TITAN is currently implemented with SCONE platform, which provides Secure Linux containers using Intel SGX and docker.

It is better to extend the compatibility of TITAN to more TEE technologies such as AMD platform Security Processor, ARM TrustZone and etc.

3, TITAN can be generalized to work with a broader set of Quality Assessment and Reputation Assessment algorithms.

Those algorithms are developed as two separated modules, which can be replaced.

4, TITAN can be evolved into a more fully featured blockchain based TIP with inclusion of the micro-payment incentive system.

The main application area of Blockchain includes financial system. Micro-payment incentive system suit TITAN perfectly.

5, The evaluation of Ethereum is based on the trust of Etherscan.

All pending transactions are fetched from Etherscan, a Block Explorer and Analysis Platform. As a consequence, the evaluation result is based on the trust of Etherscan.

To generate more concrete conclusions, running a full Ethereum Node is suggested.

Appendices

1 Pseudocode of TITAN Smart Contract

Algorithm 2 Pseudocode of TITAN Smart Contract

```
string[] userList (value: uid)
                                        ▷ uid: the userID.
string[] leaveList (value: uid)
uint[] repList (value : $R_i$)
                                        ▷ i:the index of uid, $R_i$: the reputation of i
uint[] eigList (value : $E_i$ )
                                        ▷ i:the index of uid, $E_i$: the Eigen Trust Score of i
uint[][] repMatrix (value:$R_{ij}$)
                                        ▷ i,j:the index of uid, $R_{ij}$: The reputation that provider i attains from consumer j.
mapping providerDataList (key:uid, value:List[])
                                        ▷ List[]: the dataID list provided by uid
mapping consumerDataList (key:uid, value:List[])
                                        ▷ List[]: the dataID list consumed by uid
mapping dataValue (key:k, value: v)
                                        ▷ k: the dataID, v: the quality score of k
```

```
function join(string:uid)                                   ▷ uid: the user's ID
    userList.push(uid)
    return userList.length
end function
dummy delete
function leave(string:uid)                                  ▷ uid: the user's ID
    leaveList.push(uid)
end function
function getRep(string:uid)
    $i \leftarrow userList.indexof(uid)$
    return repList[i]
end function
function getEig(string:uid)
    $i \leftarrow userList.indexof(uid)$
    return eigList[i]
end function
```

```
function updataRep(k,v,uid1,uid2)
                ▷ k:dataid, v:quality value of data, uid*: the id of user
    $i \leftarrow userList.indexof(uid1)$
    $j \leftarrow userList.indexof(uid2)$
    dataValue[k] = v
    providerDataList[i].push(k)
    ConsumerDataList[j].push(k)
    repMatrix[i][j] += v
    repList[i] += v
end function
function offloadRep(string:uid)
    if userList.exists(uid) then
        $repList[] \leftarrow repMatrix$
        Return repList
    end if
end function
function updateEig(eigTrustRes)
    $eigList \leftarrow eigTrustRes$
end function
```

2 Deployment of Smart Contract

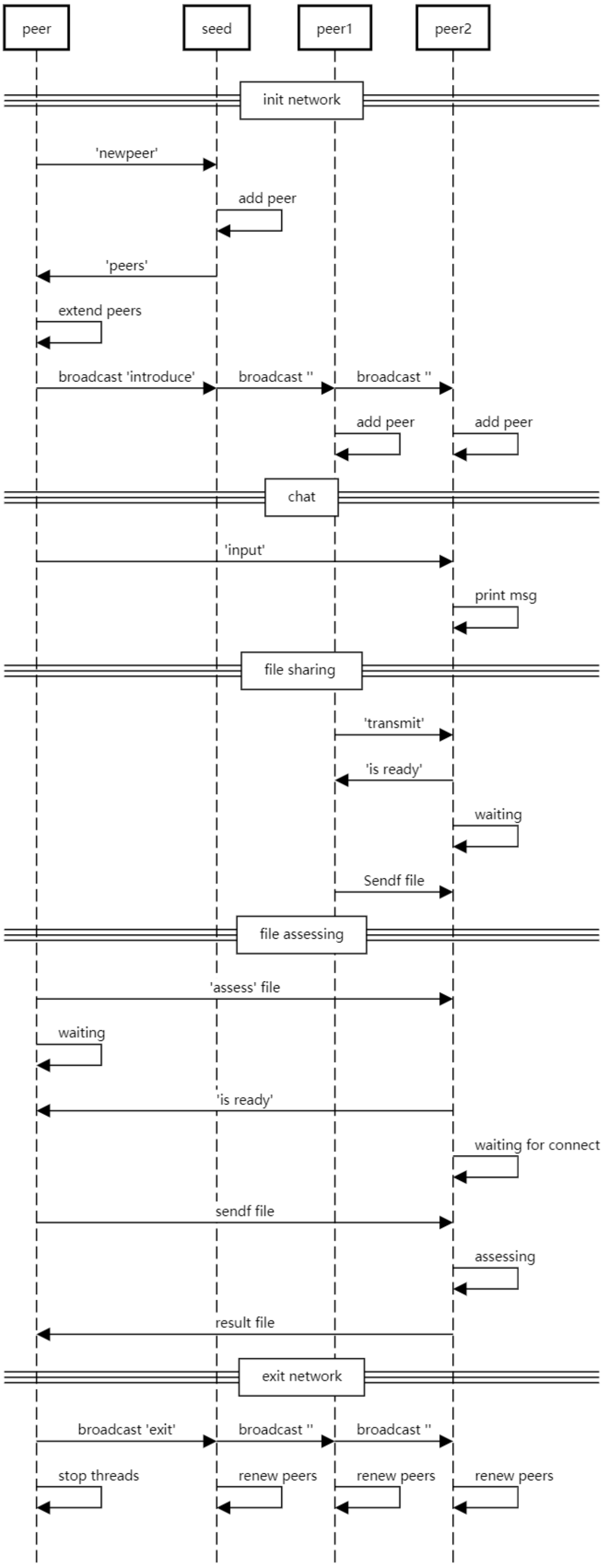
```python
from web3 import Web3, HTTPProvider
from solcx import compile_standard
import json
import os
web3 = Web3(HTTPProvider("HTTP://192.168.200.215:7545"))
base_dir = os.path.dirname(__file__)
path = os.path.join(base_dir, 'TITAN.sol')
with open(path) as f:
    data = f.read()
input = {
    "language": "Solidity",
    "sources": {
        "TITAN.sol": {
            "content": data
        }
    },
    "settings":
    {
        "outputSelection": {
            "*": {
                "*": [
                    "metadata", "evm.bytecode", "evm.bytecode.
   sourceMap"
                ]
            }
        }
    }
}
compiledSol = compile_standard(input)
bytecode = compiledSol[’contracts’][’TITAN.sol’][’Titan’][’evm
   ’][’bytecode’][’object’]
abi = json.loads(compiledSol[’contracts’][’TITAN.sol’][’Titan’
   ][’metadata’])[’output’][’abi’]

CalcContract = web3.eth.contract(abi=abi, bytecode= bytecode)
# deploy SC
list = web3.eth.accounts
tx_hash = CalcContract.constructor().transact({
    "from":list[0]
})

# get the SC address
tx_receipt = web3.eth.waitForTransactionReceipt(tx_hash)
address = tx_receipt.contractAddress

# store the abi and address for later use
data = {
    "address":address,
    "abi":abi
}
path = os.path.join(base_dir, ’abi.json’)
with open(path, ’w’) as fw:
    json.dump(data, fw)

def store(data):
    with open(’abi.json’, ’w’) as fw:
        json.dump(data, fw)

def load():
    with open(’abi.json’, ’r’) as f:
        abi = json.load(f)
        return abi
```

3 Contract Instance

```
from web3 import Web3, HTTPProvider
import os
import json
class Instance:
    contractInstance ={}
    def __init__(self):
        web3 = Web3(HTTPProvider("HTTP://192.168.200.215:7545"
   ))
        base_dir = os.path.dirname(__file__)
        path = os.path.join(base_dir, 'abi.json')

        with open(path, 'r') as f:
            data = json.load(f)


        self.contractInstance = web3.eth.contract(
            address = data["address"],
            abi = data["abi"]
        )

    def indexOf(self, uid):
        return self.contractInstance.functions.indexOf(uid).
   call()

    # join the system
    # uid is equal to the address
    def join(self, uid):
        self.contractInstance.functions.join(uid).transact({
            'from':uid
        })


    # leave the system
    def leave(self, uid):
        self.contractInstance.functions.leave(uid).transact({
            'from':uid
        })

    # update Reputation Score by data quality value
    def updateRep(self, dataId, dataValue,uid1,uid2,uid):
        dataValue = int(dataValue * pow(10, 2))
        self.contractInstance.functions.updateRep(dataId,
   dataValue,uid1,uid2).transact({
            'from':uid
        })

    # update Eigen Trust Values
    def updateEig(self, eigList, uid):
        self.contractInstance.functions.updateEig(eigList).
   transact({
            'from':uid
        })

## Read Data From SC ##
```

```
 # check the length of User List
 def getlength(self):
     len = self.contractInstance.functions.getlength().call
()
     return len

 def getRep(self, uid):
     repScore = self.contractInstance.functions.getRep(uid)
.call()
     return repScore

 def getEig(self, uid):
     eigScore = self.contractInstance.functions.getEig(uid)
.call()
     return eigScore

 def _checkEig(self):
     eigList = self.contractInstance.functions._checkEig().
call()
     return eigList

 def _checkRep(self):
     repList = self.contractInstance.functions._checkRep().
call()
     return repList

 # check the user list
 def userList(self, index):
     userList = self.contractInstance.functions.userList(
index).call()
     return userList

 # check the leave list
 def leaveList(self, index):
     leaveList = self.contractInstance.functions.leaveList(
index).call()
     return leaveList

 # return list of reputation
 def offloadRep(self):
     replist =  self.contractInstance.functions.offloadRep
().call()
     return replist
```

4 Sequence Diagram of TITAN's Peers



Sequence Diagram of TITAN's Peers

References

[1] W. Tounsi and H. Rais, "A survey on technical threat intelligence in the age of sophisticated cyber attacks," *Computers & security*, 2017.

[2] C. Sauerwein, C. Sillaber, A. Mussmann, and R. Breu, "Threat intelligence sharing platforms: An exploratory study of software vendors and research perspectives," 2017.

[3] O. Serrano, L. Dandurand, and S. Brown, "On the design of a cyber security data sharing system," in *Proceedings of the 2014 ACM Workshop on Information Sharing & Collaborative Security.* ACM, 2014, pp. 61–69.

[4] Y. Allouche, "Trusted and anonymized threat sharing using blockchain technology," 2019. [Online]. Available: https://www.first.org/resources/papers/telaviv2019/IBM-Dr.-Yair-Allouche-Trusted-and-Anonymized-Threat-Sharing-Using-Blockchain-Technology.pdf

[5] P. Inc., "Polyswarm crowdsourced threat detection." [Online]. Available: https://www.polyswarm.io

[6] C. Johnson, L. Badger, D. Waltermire, J. Snyder, and C. Skorupka, "Guide to cyber threat information sharing," *NIST special publication*, vol. 800, p. 150, 2016.

[7] N. Robinson and E. Disley, "Incentives and challenges for information sharing in the context of network and information security," 2012.

[8] O. Al-Ibrahim, A. Mohaisen, C. Kamhoua, K. Kwiat, and L. Njilla, "Beyond

free riding: quality of indicators for assessing participation in information sharing for threat intelligence," *arXiv preprint arXiv:1702.00552*, 2017.

[9] R. Whittemore and G. D. Melkus, "Designing a research study," *The Diabetes Educator*, vol. 34, no. 2, pp. 201–216, 2008.

[10] H. A. Simon, *The sciences of the artificial.* MIT press, 2019.

[11] V. Vaishnavi, B. Kuechler, and S. Petter, "Design research in information systems," 2015. [Online]. Available: http://desrist.org/design-research-in-information-systems/

[12] A. Jøsang, "Trust and reputation systems," in *Foundations of security analysis and design IV.* Springer, 2007, pp. 209–245.

[13] M. Bishop, *Computer Security.* Addison Wesley, 2018.

[14] S. Marti and H. Garcia-Molina, "Taxonomy of trust: Categorizing p2p reputation systems," *Computer Networks*, vol. 50, no. 4, pp. 472–484, 2006.

[15] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[16] N. Szabo, "Formalizing and securing relationships on public networks," *First Monday*, vol. 2, no. 9, 1997. [Online]. Available: https://firstmonday.org/ojs/index.php/fm/article/view/548/469DOI:http:/dx.doi.org/10.5210/fm.v2i9.548

[17] A. M. Antonopoulos and G. Wood, *Mastering Ethereum: Building Smart Contracts and Dapps.* O'Reilly Media, 2018.

[18] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," *white paper*, 2014. [Online]. Available: https://cryptorating.eu/whitepapers/Ethereum/Ethereum_white_paper.pdf

[19] G. Wood, “Ethereum: A secure decentralised generalised transaction ledger,” *Ethereum project yellow paper*, vol. 151, pp. 1–32, 2014. [Online]. Available: http://gavwood.com/paper.pdf

[20] D. C. Latham, “Department of defense trusted computer system evaluation criteria,” *Department of Defense*, 1986.

[21] T. C. P. Alliance, “Tcpa main specification v. 1.1 b,” 2005. [Online]. Available: https://trustedcomputinggroup.org/wp-content/uploads/TCPA_Main_TCG_Architecture_v1_1b.pdf

[22] Trusted Computing Group, “About trusted computing group,” 2019. [Online]. Available: https://trustedcomputinggroup.org/about

[23] J. C. Bare, “Attestation and trusted computing,” *University of Washington, Washington*, 2006.

[24] Trusted Computing Group, “Trusted computing platform alliance announces v.1.0 specifications for trusted computing,” 2001. [Online]. Available: https://web.archive.org/web/20030719234815/http://www.trustedcomputing.org/docs/tcpa_final.pdf

[25] Microsoft.com, “Tpm fundamentals,” 2017. [Online]. Available: https://docs.microsoft.com/en-us/windows/security/information-protection/tpm/tpm-fundamentals

[26] Trusted Computing Group, “Trusted platform module library part 1: Architecture,” 2016. [Online]. Available: https://trustedcomputinggroup.org/wp-content/uploads/TPM-Rev-2.0-Part-1-Architecture-01.38.pdf

[27] ——, “Trusted platform module 2.0: A brief introduction,” 2019. [Online]. Available: https://trustedcomputinggroup.org/wp-content/uploads/2019_TCG_TPM2_BriefOverview_DR02web.pdf

[28] Secure Technology Alliance, "Trusted execution environment (tee) 101: A primer," 2018. [Online]. Available: https://www.securetechalliance.org/wp-content/uploads/TEE-101-White-Paper-FINAL2-April-2018.pdf

[29] OMTP Limited, "Advanced trusted environment: Omtp tr1," 2009. [Online]. Available: http://www.omtp.org/OMTP_Advanced_Trusted_Environment_OMTP_TR1_v1_1.pdf

[30] GlobalPlatform, "Introduction to trusted execution environments," 2018. [Online]. Available: https://globalplatform.org/wp-content/uploads/2018/05/Introduction-to-Trusted-Execution-Environment-15May2018.pdf

[31] F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar, "Innovative instructions and software model for isolated execution." *Hasp@ isca*, vol. 10, no. 1, 2013.

[32] I. Anati, S. Gueron, S. Johnson, and V. Scarlata, "Innovative technology for cpu based attestation and sealing," in *Proceedings of the 2nd international workshop on hardware and architectural support for security and privacy*, vol. 13. ACM New York, NY, USA, 2013.

[33] M. Hoekstra, R. Lal, P. Pappachan, V. Phegade, and J. Del Cuvillo, "Using innovative instructions to create trustworthy software solutions." *HASP@ ISCA*, vol. 11, 2013.

[34] Intel, "Intel software guard extensions developer guide," 2017. [Online]. Available: https://software.intel.com/en-us/documentation/sgx-developer-guide

[35] H. Shafagh, L. Burkhalter, A. Hithnawi, and S. Duquennoy, "Towards blockchain-based auditable storage and sharing of iot data," in *Proceedings of the 2017 on Cloud Computing Security Workshop*. ACM, 2017, pp. 45–50.

[36] K. R. Özyilmaz, M. Doğan, and A. Yurdakul, "Idmob: Iot data marketplace on blockchain," in *2018 Crypto Valley Conference on Blockchain Technology (CVCBT).* IEEE, 2018, pp. 11–19.

[37] G. Zyskind, O. Nathan *et al.*, "Decentralizing privacy: Using blockchain to protect personal data," in *2015 IEEE Security and Privacy Workshops.* IEEE, 2015, pp. 180–184.

[38] K. Peterson, R. Deeduvanu, P. Kanjamala, and K. Boles, "A blockchain-based approach to health information exchange networks," in *Proc. NIST Workshop Blockchain Healthcare*, vol. 1, 2016, pp. 1–10.

[39] A. Ekblaw, A. Azaria, J. D. Halamka, and A. Lippman, "A case study for blockchain in healthcare: "medrec" prototype for electronic health records and medical research data," in *Proceedings of IEEE open & big data conference*, vol. 13, 2016, p. 13.

[40] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich *et al.*, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*, 2018, pp. 1–15.

[41] Panda Security, "Pandalabs quarterly report: Q2 2017," PandaLabs, Tech. Rep., 2017. [Online]. Available: https://www.pandasecurity.com/mediacenter/src/uploads/2017/08/Pandalabs-2017-Q2-EN.pdf

[42] D. Tosh, S. Sengupta, C. Kamhoua, K. Kwiat, and A. Martin, "An evolutionary game-theoretic framework for cyber-threat information sharing," in *Communications (ICC), 2015 IEEE International Conference on.* IEEE, 2015, pp. 7341–7346.

[43] The ISAO Standards Organization, *ISAO 300-1: Introduction to Information Sharing.* https://www.isao.org/products/isao-300-1-introduction-to-information-sharing/, 2016.

[44] FIRST, "Traffic light protocol." [Online]. Available: https://www.first.org/tlp/

[45] C. Sillaber, C. Sauerwein, A. Mussmann, and R. Breu, "Data quality challenges and future research directions in threat intelligence sharing practice," in *Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security.* ACM, 2016, pp. 65–70.

[46] J. Steinberger, B. Kuhnert, A. Sperotto, H. Baier, and A. Pras, "In whom do we trust-sharing security events," in *IFIP International Conference on Autonomous Infrastructure, Management and Security.* Springer, 2016, pp. 111–124.

[47] Protective, "Homepage of h2020 protective," 2019. [Online]. Available: https://www.protective-h2020.eu

[48] R. Sandhu, K. Ranganathan, and X. Zhang, "Secure information sharing enabled by trusted computing and pei models," in *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, 2006, pp. 2–12.

[49] L. Chen, H. Zhang, L. Zhang, S. Li, and L. Cai, "A peer-to-peer resource sharing scheme using trusted computing technology," *Wuhan University Journal of Natural Sciences*, vol. 13, no. 5, p. 523, 2008.

[50] P. Kacha, "Intrusion detection extensible alert," 2016. [Online]. Available: https://idea.cesnet.cz

[51] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in p2p networks," in *Proceedings of the*

12th international conference on World Wide Web. ACM, 2003, pp. 640–651.

[52] Truffle, "Ganache in github," 2019. [Online]. Available: https://github.com/trufflesuite/ganache

[53] S. Arnautov, B. Trach, F. Gregor, T. Knauth, A. Martin, C. Priebe, J. Lind, D. Muthukumaran, D. O'Keeffe, M. L. Stillwell *et al.*, "{SCONE}: Secure linux containers with intel {SGX}," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 689–703.

[54] Intel, "Intel sgx linux driver in github," 2018. [Online]. Available: https://github.com/intel/linux-sgx-driver

[55] PROTECTIVE project, "Protective-trustmodule in gitlab," 2019. [Online]. Available: https://gitlab.com/protective-h2020-eu/enrichment/TI-trust/protective-trustmodule

[56] L. Bach, B. Mihaljevic, and M. Zagar, "Comparative analysis of blockchain consensus algorithms," in *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO).* IEEE, 2018, pp. 1545–1550.

[57] Etherscan, "Homepage of etherscan," 2019. [Online]. Available: https://etherscan.io/

[58] Etherchain, "Homepagae of etherchain," 2019. [Online]. Available: https://www.etherchain.org/

[59] Etherscan, "Pending transactions in etherscan," 2019. [Online]. Available: https://etherscan.io/txsPending