VRIJE
UNIVERSITEIT
BRUSSEL

Master thesis submitted in partial fulfilment of the requirements for the degree of
Master of Science in Applied Sciences and Engineering: Computer Science

CLAVICHAIN

Securing the Propagation of Encryption Keys through Indirect Trust Relations

Arthur Chome

June 2021

Promotor:
Prof. Dr. Jens Nicolay
Copromotor:
Prof. Dr. Wolfgang De Meuter
Advisors:
Thierry Renaux
Christophe De Troyer

Science and Bio-Engineering Sciences

Abstract

Security in distributed systems is realized by encoding and decoding sent information with encryption keys – making the content not understandable for third parties. Opening secure channels happens by sharing the right keys with trusted parties as information encoded at one end needs to be decoded at the other end. A popular approach is to use certificates: documents binding a key to some owner which can be verified against tampering. This makes the format ideal to spread keys on uncertain networks. A node generating certificates is a *signer* and different approaches –called *models*– exist for managing signers and their certificates. Some *models* require the centralization of *signers* where nodes are recognized by the whole network to create documents. This central anchor point enables the opening of secure channels.

Networks in the context of the Internet-of-Things are dynamic and comprised of heterogeneous devices differing in available resources and used protocols. These characteristics are problematic for centralized *signers* as their resources are limited and they can drop out of the network at any time. Availability of *signers* and their services is thus limited which affects key propagation. A decentralized *model* allows for any node perceived as reliable to sign and is a better alternative for IoT networks. However, propagation is limited by the lack of available *signers* as no specialized nodes are present for the network to use. No trust propagation exists in current approach meaning only direct signers are allowed when sharing keys. As such, three parties are involved in the propagation: a key owner, an agreed *signer* and the recipient of created certificate.

In this dissertation, we examine the origin, creation and propagation of certificates to allow for multiple intermediate *signers*. A new certificate format is proposed which doubles as a record in a certificate chain with fields pointing to previous *signers*. Hash data is used against tampering. By performing a step-wise verification of the chain and its certificates up to the root, a key is validated and the chain extended. A simulation was developed in Elixir of existing and proposed *model* with the novel format. The simulation mimics distributed Internet-of-Things settings and is a way to validate our claims. We frame our contributions in current state of the art.

Acknowledgements

I would like to thank Prof. Dr. Jens Nicolay for helping me complete my end-of-study research paper. The constructive feedback sessions were instrumental in the positive development of this thesis. His guidance in past years made me learn key insights in the presentation and defence of scientific ideas for which I thank him abundantly. I would also like to thank Prof. Dr. Wolfgang De Meuter for his remarks and comments on the progress of my thesis.

Credits also go to my advisors Thierry Renaux and Christophe De Troyer for the additional support in realizing the paper. Their counseling made researching the topic much more engaging and led to some interesting developments. What stood out for me was their general patience in the course of this last year which was quite particular in some regards.

Finally, I would like to thank my family for helping me to reach this personal milestone. My parents made the venture of obtaining my Master's Degree in Computer Science a reality and their support was not limited only to the financial. My brother provided for much needed relieve next to the thesis.

Contents

1 Introduction

Security in distributed systems ensures data is only accessible to authorized parties [16]. The main challenge is the distribution of encryption keys which are pieces of information used for encoding and decoding data. All participants possess their own keys in this end. Swapping keys between nodes is necessary to open a secure channel as information encoded at the sender's side needs to be decoded at the receiver's side. A useful tool for sending keys on uncertain networks are certificates: documents binding a key to an owner with a mathematical equation to see whether its contents have been tampered with.

Secure channels are opened with either a single key or a pair of keys [4]. Schemes with two keys are referred to as asymmetric encryption [4]. One of the keys is kept a secret by its owner and the other is made public to be shared with entrusted participants. Recipients then use this shared *public* key to encode data destined to the owner who uses its own *private* key for decoding. It does not matter what key is used in a *private* or *public* role as long as one is kept a secret [43]. A problem in sending keys in order to open channels is the risk of key leakage where a third party intercepts a key which had to be kept a secret. It is less prone to occur for asymmetric encryption schemes as one key is never transmitted on a network. The security of asymmetric encryption lies in the secrecy of that key.

Asymmetric encryption algorithms are capable of creating certificates which provide a way to share keys over uncertain networks. Certificates bind participants to their keys and contain signatures for the recipient to verify against tampering. A *signer* generates certificates with its secret key and receivers verify these documents with the *signer*'s public key [49]. A recipient needs that key to perform verifications. This implies some preexisting trust relation between the receiver and *signer* as the former needs the latter's public key. A certificate contains data on the *signer*, an encryption key and its associated owner [2].

Certificate-based models either assign nodes as recognized *signer* to make certificates for the whole network or allow any node to do so which makes the model decentralized [53]. The latter approach is referred to as Web of Trust and Pretty Good Privacy (PGP) is a notable example of its implementation [42]. Opening channels happens with any intermediate signer trusted by the key owner and recipient [2]. *Trust* is therefore defined as the degree in which a node is deemed reliable to function as signer. In a scenario where Alice wants Bob's public key to open a channel,

a third party Dave is used to swap the keys – Dave is trusted by Alice and Bob. Dave signs a certificate on Bob's request binding its encryption key to the document. Alice retrieves Dave's new certificate and assumes the key to be of Bob based on a successful verification and the trust bestowed on her friend Dave.

At most one signer in between parties is allowed in Web of Trust and this constraint is imposed to limit the falsification of certificates. A node of which no relation is indicated, is assumed to be not trusted. Relations between nodes are unique for each and global parameters are applicable to all. Acceptance parameters determine the amount of certificates to collect before accepting associated key. Multiple signers need to provide a certificate in order to reach the amount of certificates to collect. A way to improve confidence in the collected certificates is by increasing the acceptance parameter though this negatively affects propagation. More certificates are to be collected in order to accept a key.

A novel trend in the field of connectivity is the Internet of Things[40] referring to networks established from seemingly day-to-day "things"[48] able to receive and send data on their own, blurring the line with normal environments. It encompasses surrounding objects of a human's living space such as home appliances, machines, transportation, business storage, etc. *Smart* devices are fitted with embedded systems as to interact seamlessly with their environment. Centralization in Internet-of-Things makes a network more manageable with the presence of supervisors [35]. Web of Trust has some shortcomings in its model hindering its applicability in IoT settings. A possible way of improving key propagation is by studying the origin of certificates. It is related to the field of *provenance*, which studies the "problem of detecting the origin, creation and propagation of data" [10].

1.1 Problem Statement

Web of Trust as a decentralized trust model has been successfully applied to email encryption with the PGP software. Secure channels are opened between parties with certificates signed by mutually trusted intermediates. The notion of trust is statically configured at initialisation and makes the model more difficult to apply in dynamic networks such as the Internet-of-Things. Participants in such networks can join and leave as they please which undermines the availability of signers. Nodes lack the necessary certificates when signers are unavailable due to network changes. Web of Trust displays following shortcomings:

- **No Mechanisms for Trust Propagation** direct trust relations between nodes are indicated at initialisation. Indirectly connected nodes do not necessarily trust each other as the notion of trust is not transitive. A key owner trusts its direct intermediates but none that come after even if these parties have the same direct connections. Spreading a certificate beyond the first intermediate is not possible as no mechanisms exist to spread trust to these additional signers.

  Trust is mutual but not transitive, meaning a direct trust relation between two participants does not imply a relation with indirect participants. No mechanism exists in the Web of Trust to extend trust to indirect associates.

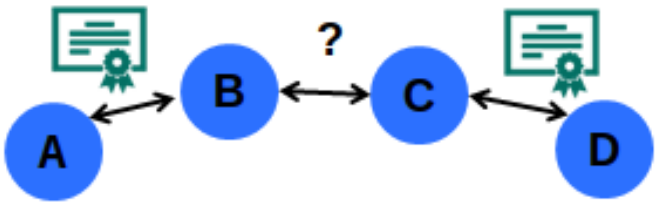


Figure 1.1: Node A, B, C, D form a linear trust relationship. Node A will not have access to D's certificate in Web of Trust because it trusts B as signer but not the nodes that B itself trusts as signer (node C). A mechanism needs to be developed in order to propagate trust between signers.

- **Only One Signer in Between Parties** allowing at most one signer is a constraint imposed in Web of Trust to limit falsification. It also flows from the lack of trust propagation. Relaxing the constraint improves key propagation but increases the risk of forgeries. A receiving node does not trust all previous signers and no mechanism exists to verify the whole chain. Because indirect signers cannot be verified, indirect trust is not possible. Participants in the Web of Trust are not required to keep track of received certificates which complicates a possible verification to take place.

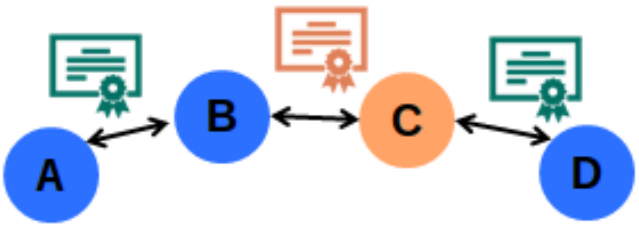


Figure 1.2: Relaxing the constraint of one signer by allowing for a sequence of signers increases the risk of injecting false keys. Take the example of previous point where nodes A, B, C and D are linearly trusted and sign each others' certificates. If a node A receives the certificate of node D from the intermediates in between, it has no way of trusting node C's certificate which is one intermediate away. No mechanisms are in place to verify signer-of-a-signer certificates.

The current certificate format of Web of Trust has no bookkeeper information describing the origin of a key and its previous signers.

1.2 Approach

Our dissertation's main goal is to extend the Web of Trust model with multiple signers for certificate propagation. By allowing for transitive trust to occur, participants with no direct connections trust each other. Chains of signers lead to more keys being passed and more secure channels to be opened in comparison with intermediate signers. A simulation was developed of this approach and Web of Trust to measure the effects on key propagation.

With a newly developed certificate format, it is possible to introduce signer-of-signer relations in the model. Provenance information included in the format allows a verification of the whole chain of signers against tampering. A signer extends the *chain* by generating a new certificate from previous one. A new upper chain length parameter specifies the highest allowed number of signers preventing a forger to indefinitely redirect the verification process to collaborating nodes.

To summarize, we want to allow for multiple intermediates in the propagation of certificates in order to open more secure channels. Global acceptance parameters in both approaches remain the same. This makes the model more applicable in Internet-of-Things settings where intermediate signers drop in and out dynamically.

1.3 Contributions

Adding provenance to the certificate format allows for multiple signers to be linked together to spread keys. The number of opened channels increases in the Web of Trust for the same global parameters or the amount of certificates to verify before accepting a key. Our dissertation presents the following contributions:

A Novel Certificate Format based on Provenance it is possible to link multiple signers together with a new certificate format. PGP originally does not allow for more than one intermediate on accounts of the risk of forgery but added provenance allows for a verification of the whole chain. A participant extends the chain of certificates by creating its certificate from an original document. In turn, another node extends the chain based on that document. Immediate effects on security of this new format are an increased propagation of keys. No longer is the integrity of a certificate based upon the trust relation with the signer. If an intermediate injects a false key in the certificate, this is detected in the previous chain of documents.

Simulation in Elixir an implementation was built in Elixir of proposed approach and existing Web of Trust model. The latter resembles the existing PGP software for email encryption. Assessments on the integrity of certificates are improved with provenance and the following metrics measure its impact: coverage, spread and accuracy [29].

Thorough Comparison of Provenance-Based Approach against Web of Trust a set of benchmarks and scenarios were compiled to measure the properties of approaches. Results from evaluation show how certificate chains improve key propagation. More secure channels are opened between participants and forgery is more accurately detected compared to existing Web of Trust model. A downside of our approach is increased resource consumption as related work stated that making a trust model more robust also makes it more complex [6]. In addition, the propagation of keys tends to stagnate for longer chains as verification time grows with the number of participants.

1.4 Overview

The remainder of the paper is structured as follows:

Chapter 2: Context high-level explanation with a use case involving an Uber-like taxi service. General concepts are explained such as the Internet-of-Things, the need for security, encryption, certificate-based trust, etc.

Chapter 3: Related Work a literature study describing the state of the art of current research to frame our contributions. Where alternative techniques and research paths exist, justifications are provided on which one was selected.

Chapter 4: Approach overview of the solution and how it answers the research goals cited in section 1.3 of this Chapter. The certificate format with provenance is explained more in detail. Some research efforts explored policies for chain reparation and are briefly discussed.

Chapter 5: Implementation overview of the simulation's development such as how the proposed model is implemented against the PGP model in terms of message flow, etc.

Chapter 6: Evaluation an empirical validation of proposed model. By tweaking the simulation's parameters, we show the format's performance at runtime as well as resource consumption.

Chapter 7: Conclusion based on the problem statement in section 1.1 and evaluation results from Chapter 6, we state our findings. Provenance makes for higher resource consumption for basic operations but improves coverage, spread and accuracy metrics in a significant way. The coverage and spread metrics are correlated with each other and tend to the same values for bigger networks. Keys are propagated further but the marginal gains tend to diminish for longer chains attributed to the longer verification times – more certificates. Falsification is more easily detected with provenance as the whole sequence of linked documents is concerned. Web of trust is made more robust against malicious parties and resource consumption increases as a trade-off.

2 Context

This chapter focuses on the technologies used as a starting point of this dissertation. We provide a taxi service use case as an illustration of identified problems. Section 2.1 shows the characteristics of networks belonging to the Internet-of-Things, while section 2.2 presents an Uber-like taxi service with a centralized architecture as a real-life use case of IoT. Its limitations are discussed and section 2.3 proposes a decentralized architecture as a solution better suited for dynamic networks. It relies on Web of Trust to open channels which has some problems in IoT environments. This sets the stage for our approach in Chapter 4.

2.1 Internet of Things

A novel trend in the field of connectivity is the Internet of Things [40] referring to networks established from seemingly day-to-day devices able to receive and send data. As of the time of writing, the field encompasses surrounding objects of a human's living space such as home appliances, machines, transportation, business storage, etc. *Smart* devices are fitted with embedded systems for seamless connectivity in their environment [40].

2.1.1 Aspects of Security

A network needs to be robust enough to remain functional even in the presence of malicious nodes attempting to access confidential information or pass incorrect keys to their peers. Security is described in terms of secure communication, authentication, access control, provenance and trust [28]. We provide a short summary on their description in this section.

Secure communication prohibits unauthorized parties to access information over a network. An encryption scheme is used which takes as input an original message and an encryption key in order to generate a seemingly random mesh of characters. The resulting sequence is sent over the network and incomprehensible for parties not in possession of the right keys. Sharing decryption keys with peers grants them access to the data and opens new secure channels [16]. Proper key management and propagation is essential for secure communication.

A method to authenticate participants needs to be present to complement secure communication in order to identify the participants to share keys with. Parties claiming some identity need to be verified for said identity in order to determine whether keys are to be shared with them [27]. Some resources are limited and parties granted access need to be identified. Access control deals with restricting resources to a chosen few [19]. Certificates unlock capabilities on the network such as accessing the content of encoded messages. Provenance solves the "problem of detecting the origin, the creation, and the propagation processes of data" [10] and it is leveraged to detect forgery on certificates shared with parties to access the content of messages.

Trust is a concept active in different scientific domains with related work proposing multiple definitions [6]. An approach in determining trust with participants is referred to as a *model* and literature proposes either behaviour-based or certificate-based *models* as an approach to trust [7]. Behaviour-based trust is described as "a subjective expectation an agent has about another's future behavior based on the history of that agent, and other reliable agents' encounters with that other agent" [39]. It is the esteem in which a node is regarded in terms of reliability determined by experience, recommendation and reputation.

Behaviour-based approaches are not popular in the field of IoT for different reasons. Node esteem needs to be updated continuously which is quite a resource overhead on already constrained nodes [6]. Behaviour-based models do not have a scheme for key propagation included which is already provided with certificate-based models [6]. A certificate is a document serving as proof-of-ownership to a key with a signature against tampering. It allows to open secure channels with trusted participants in a capability-based manner.

2.1.2 Challenges

Any device capable of emitting and receiving data through dynamic wireless connections is able to participate in IoT networks [6]. Embedded systems are more constrained in terms of resources and some security technologies used for traditional personal computers do not work in this context [48]. Modern computers are expected to have a minimal compatibility towards each other in terms of used protocols and available resources and this complicates ensuring security for embedded systems [28]. Assessing scalability is also more complex as workloads are more difficult to allocate on devices with different characteristics [28].

IoT networks are dynamic in nature and typically have limited space for centralization [6]. A lack of centralization makes the network more fragile as no settled authority exists to regulate the system. A benefit associated with decentralization is flexibility as bottlenecks are less prone to occur in systems [6]. Bottlenecks in distributed systems occur when too many requests are made to some service-providing party, slowing down its responsiveness [7].

Another issue is the lack of conventions and guidelines when developing IoT applications. Involved devices differ in terms of used protocols and available resources. Finding common interfaces remains a challenge in IoT and the field remains novel in the research community [54].

2.1.3 Smart Environments

A device is considered *smart* if it is capable of obtaining and applying knowledge independently [5]. It is fitted with embedded systems to sample and send data over a network. *Smart* devices offer new opportunities to improve work flow in existing industries by way of device cooperation [11]. IoT has been considered in sectors such as smart cities where sampled data is used to benefit the way of life of inhabitants and businesses [55]. Another application is in the transportation sector by fitting mobile vehicles with sensors to improve task performance [5].

2.2 Centralized Cars

Section 2.1 discussed the security aspects and challenges of the Internet-of-Things. In this section, a centralized taxi service is presented as a use case of IoT with its architecture requiring a data centre in each service-providing area. It is similar to Uber[1] in its operation: an internet-based mobile application is used to connect customers with their drivers and the data centre. A customer specifies a taxi's rendezvous point for the system to dispatch a driver to.

2.2.1 Certificate-Based Trust

Section 2.1.1 explains how approaches to trust are either behavior-based or certificate-based. A certificate is a document binding a key to some owner with a signature to control its integrity. A certificate is passed on the network and verified against tampering at the receiver end. IoT settings are better suited for a certificate-based approach as certificates require less resources to be used and their way of working already provides a mechanism for key propagation. Behavior-based approaches rely on reputation and experience to open a channel but have no included method for key propagation and require additional resources for continuous updates [6].

2.2.2 Certificate Authorities

The centralized architecture of the taxi service uses a data centre for each region and its mobile application requires both customer and driver to register an account within that region. The data centre uses certificates to share encryption keys to the drivers. It would function as a Certificate Authority (CA) for the area, creating or *signing* certificates to open channels with registered customers and drivers. Every new region in which the service is expanded needs a data centre to coordinate drivers to their customers [20]. It makes for added stability in the network as a synchronization point for shared locations, payments, etc.

2.2.3 Limitations

A centralized data centre in each service-providing area coordinates taxi vehicles to their clients and is a central anchor point to manage location information and payments. Leveraging centralization makes for a stable network with appropriate security levels. Some shortcomings were identified in current architecture when it is applied in smart environments.

The necessity of a data centre makes the service more difficult and costly to deploy in areas lacking the required infrastructure. Allocating roles to network participants causes potential system bottlenecks [6] and the specialized signers are points on which dedicated cyber-attacks are performed.

[1] `https://www.uber.com`

2.3 Smart Cars

We now propose a similar Uber-like taxi service but with a completely decentralized certificate-based approach on trust. By fitting involved taxi vehicles with embedded systems, information normally stored and processed in the region's data centre is managed individually. Limitations of a centralized architecture mentioned in section 2.2.3 are alleviated as no specialized roles exist in the network. Security tasks are handled individually eliminating the obligatory data centre in previous architecture: the service is easier to deploy in new areas.

2.3.1 Web of Trust

Previous architecture required a data centre to open secure channels by sharing certificates. Web of Trust is a fully decentralized approach to trust and has no recognized *signers* to create or rather *sign* new certificates. A vehicle is responsible for opening secure channels with its peers and signs its owns certificates sent over uncertain networks [31]. Involved *smart* vehicles function as intermediates between a key owner and receiver.

Trust Relations

Within Web of Trust and its implementation PGP, a node deemed trusted by two parties is placed as an intermediate to swap keys. It assumes a *signer* role and creates the certificates as a proof-of-ownership. A receiver assumes the key to be of the owner in question on grounds of its trust relation with the intermediate and on its successful verification. Web of Trust and its implementation PGP works by way of trust relations between peers settled at pre-deployment time. A participant indicates which node is trusted to sign and a tagged node is either *fully* or *marginally* trusted in the model [2]. Participants check their trusted neighbours for new certificates. A key is accepted if enough certificates binding its owner are successfully verified.

Accepting Certificates

A system is initialized with acceptance parameters which determine how much certificates are to be collected by a party in order to accept a key associated to an owner. A node needs a different number of certificates before accepting based on the parameters applicable to all participants [2].

2.3.2 Limitations

A decentralized taxi service removes the need of a data centre and makes it applicable in areas lacking appropriate infrastructure. Web of Trust as a trust model has some shortcomings in smart environments which are to be addressed in this dissertation.

Dependence on Trust Relations

A decentralized architecture is more flexible but also more fragile as no authority figures are there to regulate the system [6]. Web of Trust is dependent on the nodes indicated as trusted *signers* to share certificates in order to open channels. An assumption made by the model is that trusted nodes are indeed reliable. If a participant betrays the trust bestowed on him by committing forgery on certificates, no mechanism exists to detect the malicious act.

Acceptance Parameters and Skepticism

Setting the acceptance parameters sufficiently high mitigates the effect of forgers as more certificates are needed to accept a key. More certificates are verified in order to detect inconsistencies in sampled documents. Enough documents have to be successfully verified. Increasing acceptance parameters are a way better detect inconsistencies or forgeries at the expense of propagation. Web of Trust and its PGP implementation tend to be less effective in more adverse environments.

Lack of Intermediate Chains

PGP has a strict requirement that only one intermediate may be placed between a key owner and an eventual receiver – meaning only one signer partakes in the transaction of certificates. No signer-of-signer relations are possible and there is no way in verifying the sequence of participants. Web of trust and its PGP implementation are said to lack trust propagation mechanisms.

2.3.3 Visualization

Let us consider a fully decentralized network of *smart* cars driving customers to their destinations, similar to Uber and with no data centre. Cars have a key pair to encrypt and decrypt sent information and channels are opened by passing the correct keys to trusted receivers.

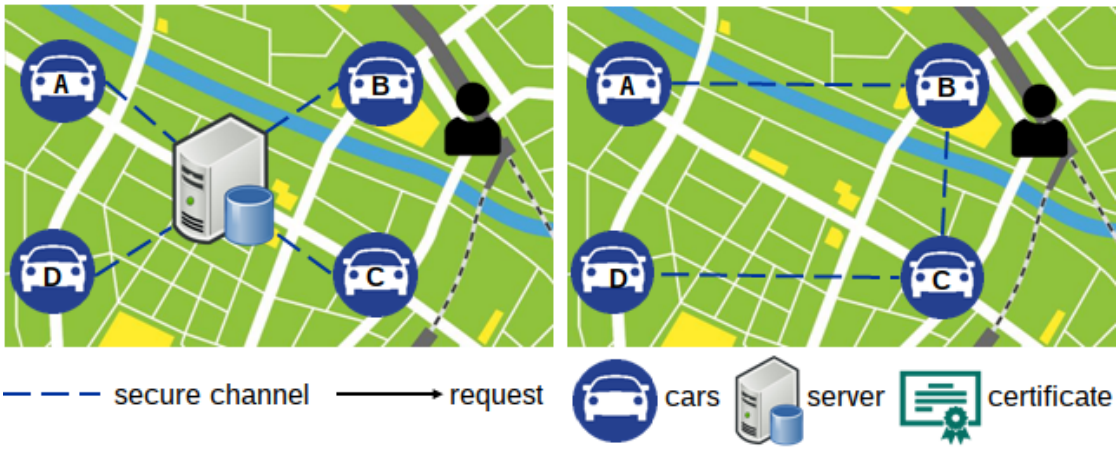


Figure 2.1: Panel left shows the architecture of the Centralized Cars application. The server coordinates the communication between cars and the customer. Right panel shows an equivalent service without a centralized server. Blue lines indicate a *signer* relation between cars.

Smart Cars: Successful Propagation

Involved cars need to manage security themselves in a smart environment. Cars make use of Pretty Good Privacy to pass encryption keys and the acceptance parameter is set at one. Collecting one correct certificate from a trusted issuer suffices to accept a key.

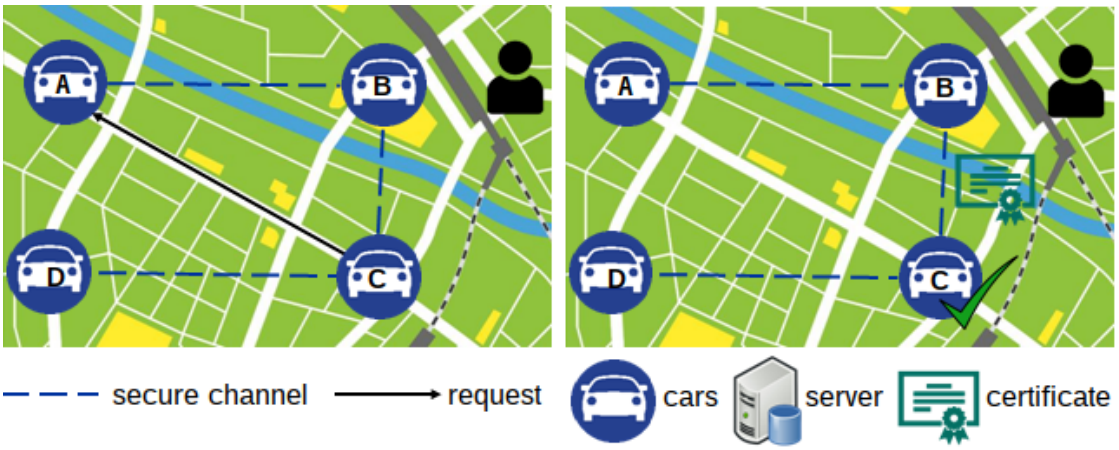


Figure 2.2: With trust being mutual in our definition, we can see how the cars have a linear trust distribution – A trusts B, trusts C and trusts D. All of them possess each other's keys that were physically traded at pre-deployment time.

Opening additional channels happens by sending certificates to each other using intermediate signers. If car C wants to open a channel with car A, it needs to send a request to the owner node in question. Car A swiftly responds by *pushing* a request to trusted car B to sign the certificate and display it for other nodes to notice. Car C will check the displayed documents of B and *pull* the new certificate as to accept its key after verification.

Smart Cars: Failure Case

Previous scenario showed how certificate propagation in PGP works well if one intermediate is present. If the requirement is not fulfilled and multiple intermediates are required, the key will never reach the node in question [31]. Propagation worsens when trust relations are lacking.

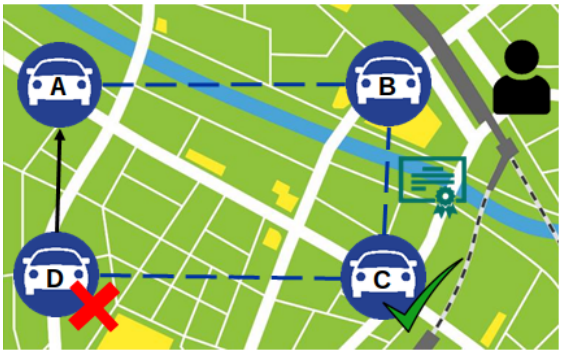


Figure 2.3: Car C will have the requested key provided by a direct intermediate but car D will never receive it because it lacks an appropriate issuer. Intermediate chains –issuer-of-issuers– are not possible in PGP.

Smart Cars: Naive Approach

A possible solution to previous scenario would be to relax PGP's requirement of direct issuers – signers adjacent to the owner and receiver. Nodes not in direct contact with the owner can serve as signer to spread the certificate.

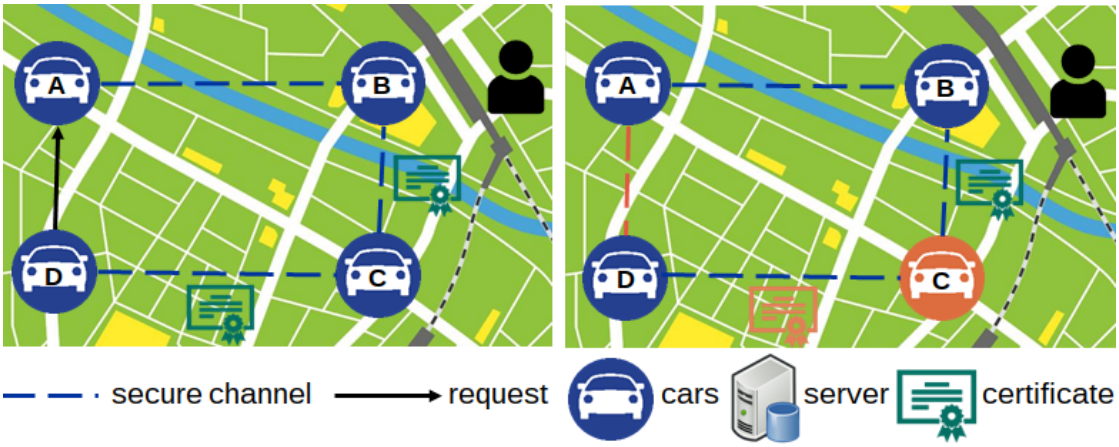


Figure 2.4: If the constraint of one signer is relaxed, the risk of forgery by the second intermediate is too high as trust is not propagated through signers. C injects a wrong key in its certificate to recipient D. No methods are in place to verify the sequence of signers.

In practice, such approach is not feasible. The risk of malicious issuers compromising the chain with incorrect keys is too high.

2.4 Conclusion

In this chapter, we discussed the Internet-of-Things and its characteristics in terms of challenges and security as a prerequisite for smart transportation applications. An Uber-like taxi service is discussed using a centralized and decentralized trust model. The latter is referred to as the Web of Trust and some limitations were identified when applied in smart environments. Its reliance on good-behaving nodes for certificate distribution, failure to detect forgeries and lack of intermediate chains makes it unsuitable in smart transportation applications. Beefing up the acceptance parameters on a network negatively affects propagation while relaxing some of its constraints increases the risk of forgeries.

3
Related Work

This chapter is a literature study to help us frame our approach in current state of the art. The collected papers cover topics on security in dynamic networks. A justification is provided on the chosen technologies when alternatives exist. Our solution's design reflects our assumption that the target system is part of the Internet-of-Things. We refer to the smart cars taxi service from Chapter 2 as a reference. We refine our research questions at the end of this chapter if necessary.

3.1 Secure Communication

Secure communication relies on encryption keys to encode and decode data. They are pieces of information which are used in combination with an encryption scheme to take some original message and translate it to some *ciphertext* which is incomprehensible without the proper keys. A secure channel is opened with a party when it detains the necessary decryption keys to translate the *ciphertext* back to its original message upon arrival. A distinction is made between symmetric and asymmetric encryption [30] with Web of Trust and its implementation PGP relying on the second to open channels. Asymmetric encryption algorithms use two keys with one of them being kept a secret by its owner and the other made available for the wider public. Recipients use the *public* key to encrypt data for the owner who uses its *private* key for decryption. It does not matter what key is used in a *private* or *public* role as long as one of them is kept a secret [43]. The unwanted leakage of a key to third parties is less prone to occur with two keys as one of them is never sent over a network. Asymmetric encryption's strength lies in the secrecy of the *private* key. Resource requirements are an important factor when choosing the algorithm as well as its robustness: how difficult it is for a party to *crack* the cipher given some encoded message [4].

3.1.1 Symmetric Encryption

Using one key for both encryption and decryption is advantageous in terms of simplicity and efficiency [4]. Problems arise when propagating keys as the unwanted leakage to third parties is much more likely to occur. A pair of keys has the advantage of never sending one of its keys over the network. A single key for encryption and decryption is not the preferable option

for the development of public networks [45] as certificates cannot be signed in this category. Symmetric algorithms are unusable for certificate-based trust models [8] and we therefore dismiss the category altogether.

3.1.2 Asymmetric Encryption

Algorithms of this category use one key to encrypt messages and another to decrypt them. Former key is available to the public whereas the latter is kept private. At least one of the keys must be kept a secret for the scheme to work. Asymmetric algorithms do not have the same propagation issues of their symmetric counterparts as one key is never sent over the network. Algorithms of this type can generate signed documents for propagation and key leakage is less of a problem with the public-private key distinction [33]. Higher resource costs are associated with two keys and it is not recommended for networks with devices with very low resources [44]. Some papers claim symmetric algorithms to be 1000 times faster in comparison with asymmetric algorithms [23]. Different algorithms were identified as candidates for our smart cars application.

Rivest-Shamir-Adleman

The Rivest-Shamir-Adleman (RSA) cipher was proposed in 1978 and is one of the most popular asymmetric encryption algorithms in use [43]. Its strength lies in the factoring problem of mathematics [9]. It is speed-wise slower than symmetric algorithms and demands a considerable key size to reach adequate security levels [22]. Sensor networks are perceived as too constrained for the RSA algorithm [46]. It thus demands quite some resources in order to be applied.

Elliptic Curve Cryptography

ECC relies on the intractability problem of discrete logarithms [32] to generate shorter keys than in RSA for similar security levels [21]. Elliptic Curve is a viable option for constrained sensor networks when RSA encryption of previous paragraph is too demanding [6]. For ad-hoc networks like the smart cars case, using RSA or Elliptic Curve is not a significant consideration due to the availability of resources.

Diffie-Hellman Exchange

Diffie-Hellman allows for two parties to share each other's secret symmetric keys without compromising the network. Involved parties agree on a publicly shared number and a privately held secret number. By adding this number to the public value and sending it to the other party, both end up with the same number representing the secret key to be used. The protocol is seen as insecure for well-funded attackers such as governments or federal institutions [3].

3.1.3 Summary

Drawbacks of symmetric encryption schemes are the difficulty of key propagation with key leakages being a significant problem [13]. Asymmetric encryption algorithms are able to sign certificates, providing a way of sharing keys on uncertain networks. Asymmetric encryption might not have the fastest performance but a trade-off is higher network security due to the secrecy of at least one of its keys [22]. RSA is used in the code simulation because of its popularity and also due to the availability of packages in Elixir. Elliptic Curve could have been a more cost-effective alternative [32] but the code base lacks the necessary packages. Its implementation was not deemed necessary for the progress of this dissertation.

3.2 Definitions of Trust

A concept existing in different fields, related work proposes multiple definitions [39]. Approaches in determining trust are either behavior-based or certificate-based. An approach is also called a *model*. Within a behavior-based model, trust can be defined as "a subjective expectation an agent has about another's future behavior based on the history of that agent, and other reliable agents' encounters with that other agent" [39]. Certificates open up secure channels in a capability-based manner and are shared with nodes perceived as trusted. Nodes are given the capability to securely communicate.

3.2.1 Behavior-Based Trust

Trust in this model depends on experience from post interaction, reputation from the opinions of trusted nodes perceive and knowledge [12], which is a cumulative value representing past interactions. Computation of trust is done independently by each node and requires continuous monitoring and exchange of recommendations [8]. Monitoring is a resource-draining task too costly for sensor networks to handle [8]. Requirements for monitoring increases with the strain of messages put on the network [6]. It is believed not to be the best option for smart environments due to its resource-demanding nature. An additional problem is the propagation of keys. Behaviour-based approaches require a dedicate mechanism for key exchange [3].

3.2.2 Certificate-Based Trust

Certificates are a mechanism for key propagation. One can assume a sent document is unaltered after verifying its signature. Certificates are used in a capability-based manner [34] where owning the right documents enables secure communication with peers. Nodes send requests to owners and if judged trustworthy, certificates are propagated to open new channels. Using certificates solves the issues previously underlined. Continuous monitoring of messages passed on the network is not necessary as trust is bound to its certificates. Propagation of keys is included within the passed certificates as the signatures are used as an integrity check.

3.2.3 Hybrid Approach

A combination of both behaviour-based trust and certificates is possible [18]. It builds on PGP as being a decentralized certificate-based model which leverages the experience of a node's trusted neighbourhood to make better decisions on whether to accept certificates. Giu's paper [18] showed an improved version of PGP by considering the recommendations of trusted participants. It leverages the power of behaviour-based trust to make better decisions whereas this dissertation relies on deeper provenance-based verification. The metrics proposed in the paper were of significant influence on the final evaluation of Chapter 6 [18].

3.2.4 Summary

Certificate-based trust is less resource-demanding as no continuous stream of updates need to be passed on the network. What's more, a certificate on itself provides a way to propagate keys over uncertain networks. Certificates are much more appropriate for the development of Chapter 2's taxi service due to them being less complex and resource-demanding.

3.3 Architecture

A certificate-based trust model can be implemented in different ways. A trusted node can be assigned with a supervisory role for signing or it could be done in a decentralized way in which any participant creates certificates. Hence, any party is able to be *signer*. Networks manage trust either by relying on an authority figure or by computing trust on their own. A trade-off needs to be made when choosing and this section explains why a decentralized approach is the best option in developing a taxi service in the Internet-of-Things.

3.3.1 Public Key Infrastructure (PKI)

A PKI specifies a set of roles and policies to properly manage certificates and keys within a network. A globally recognized signer node is called a *Certificate Authority* (CA) and assists in the propagation of certificates by serving as intermediate [38]. Certificate Authorities follow a hierarchical structure where parent nodes certify child nodes to serve as recognized signer. Revocation happens by the top-most signer in the hierarchy [36]. Having a centralized node to manage certificates makes it a system bottleneck [6]. It is visible for dedicated attackers and temporary certificate authorities can be assigned to reduce the problem of attacks [27].

3.3.2 Web of Trust

Aivaloglou [8] proposes an architecture where each node would take on the role of a certificate authority passing certificates to nodes that have been deemed reliable in the network. Certificates are passed to nodes with sufficient trustworthiness and can be revoked if deemed necessary. What we are referring to here is the Web of Trust where any node can be the intermediate signer to pass a certificate. Its most popular application is PGP which specifies that at most one intermediate may be used in the transaction of a certificate between owner and requester [2].

3.3.3 Summary

Public Key Infrastructures (PKI) assign some roles to nodes in order to provide services to their peers such as certificate signing. A disadvantage here are system bottlenecks [6]. It is better to avoid dedicated Certificate Authorities in dynamic networks as it undermines flexibility. A certificate-based decentralized approach seems to work best for IoT and its properties [2].

3.4 Data Provenance

Data provenance is concerned with the "problem of detecting the origin, the creation, and the propagation processes of data" [10]. What matters is the preservation of security as one of its applications it the validation of reliable participants [24]. The field can be leveraged to improve key propagation and we describe the found technologies that are candidates to achieve this goal.

3.4.1 Hash Chain Scheme for Provenance

Suhail's paper [47] proposes a hash chain scheme where data packets carry an additional tag for data provenance. A node adds a hash of its id to the tag upon receiving a packet and forwards it to the next node. Verification happens at the sink node by controlling if each hash belongs to a trusted node. Although it works well to determine the origin of a packet, no mechanisms are provided to assure integrity. Performance of this approach becomes worse over time as the chain becomes longer. Purging the chain at some length can improve its performance.

3.4.2 Blockchain-Based Provenance

Blockchain is a tamper-proof distributed ledger that manages and validates some past transactions [56]. There are different consensus mechanisms for its implementation [17]. Javaid's paper [26] proposes an implementation of blockchain in combination with physically unclonable functions to realize provenance. IoT devices need to register as to transfer data by interacting with the smart contracts which communicate with underlying blockchain. Although provenance is hard to tamper with when leveraging the power of blockchain, it seems quite complex for the constrained nature of smart environments.

3.4.3 Public-Key Linked Records

Wang's paper [51] suggests a provenance scheme that could be used to spread information over multiple cooperating devices. Data forms a provenance chain with each provenance record being stored in one device. Cumulative hash functions are provided for integrity verification. It is possible to reconstruct the totality of the data with the pointers to previous devices. It also includes a signature per record for verifying the fields themselves. A device's public key is included in its neighbours' records in order to enable them to verify the device's record.

3.4.4 Summary

We gathered a collection of papers for data provenance. The hash chain scheme of section 3.4.1 builds a provenance chain to reconstruct the origin of data. Blockchain-based data provenance requires quite some resources to be realized which is too demanding for the smart cars application of Chapter 2. Public-key linked provenance records presented in 3.4.3 seems to be the best choice for provenance between certificates. It contains both cumulative hashes to verify the previous members and signatures to verify current record's integrity. Wang's paper [51] was influential in the design of new certificate format of Chapter 4 and the evaluation metrics of Chapter 6.

3.5 Conclusion

This chapter discussed possible technologies and their alternatives to secure dynamic networks. Section 3.1 explained the benefits of using asymmetric encryption in dynamic networks and how two keys make for a more robust encoding of data [33]. A trade-off was the steeper resource consumption and increased execution times compared to algorithms part of symmetric encryption [44]. Key leakage is a situation in which an unauthorized third party intercepts a shared key and compromises safety. Leakage is less prone to occur with asymmetric algorithms as one of the two keys is never transmitted over a network. It was decided to go with asymmetric encryption for its additional safety at the expense of resource consumption [22].

Asymmetric encryption allows for the creation of certificates. An owner has its key associated to a certificate and a receiver verifies its contents with the provided signature. In doing so, falsification over the network is detected. Section 3.2 discussed certificates as effective in propagating keys and as an approach to trust [34]. Parties viewed as trusted are to receive a certificate in order to initiate secure communications. Certificates are used in a capability-based manner. Behaviour-based trust models have no included key exchange mechanism in their way of work and display additional overhead in the propagation of recommendations [3]. Behaviour-based trust is ill suited for use in the Internet-of-Things and its constrained nature.

Section 3.3 discussed possible architectures for certificate-based trust models. Either specialized *signers* were used or any node was allowed to make certificates. Centralization makes networks more stable but it was considered not a good fit for the dynamic nature of IoT systems [6]. Web of Trust is seen as a better architecture in IoT, even with its existing limitations.

Section 3.4 gave an overview of techniques for provenance to trace back the origin of data. Technologies ranged from a hash chain scheme [47] to blockchain technology [26] and it was decided that public-key linked records [51] are a better technique for provenance. It includes a hash scheme and signatures to verify documents. It made the switch to the development of a novel certificate format with provenance even more apparent.

4 Approach

In this chapter, we describe a novel certificate format linking documents together into one coherent chain. A certificate doubles as a record which is part of a whole sequence of documents with verification of a certificate being extended to the sequence. The goal of our approach is to make the Web of Trust model more suited in dynamic networks by increasing key propagation. We list the metrics used to measure the impact of the proposed format. It helps us reason about the advantages and disadvantages of linking certificates together.

4.1 Overview

Web of Trust is a certificate-based trust model on itself fully decentralized: any participant is able to create certificates if it is trusted by the two key-swapping parties. Goals of our approach are to make the Web of Trust model more suited for its use in dynamic networks. Our approach is named *Web of Chains* and its format *ClaviChain* links certificates as records together in a sequence. The mayor parts of our approach are as follows:

- **Web of Trust** the original approach on trust is summarized. Its implementation is referred to as *Pretty Good Privacy*. A certificate is created by a *signer* and only one node is set between two parties to take on a *signer* role. Details include the propagation of keys in terms of push and pull operations to be compared later with propagation in Web of Chains.

- **Web of Chains** a variation on the Web of Trust model with a new certificate format. It has pointers to previous *signers* linking certificates together in a chain. Its implementation is referred to as *ClaviChain*. It has back pointers to perform a step-wise verification where certificates are fetched and verified up to the key owner or *root* of the chain.

- **Discussion** significant efforts were put in policies for ClaviChain to deal with inconsistencies in certificate chains and the selection of candidates to extend a chain. It is mentioned here although not pursed in later chapters.

- **Metrics** after having discussed the propagation models of Web of Trust and Web of Chains, penultimate section deals with the metrics for measurements. Insights on the models' performance are obtained by using said metrics.
- **Simulation** as a prelude to next chapter, the simulation is described in terms of its alterable parameters such as trust relations, etc. More details on the simulation and its implementation are provided in Chapter 5.

4.2 Web of Trust

A Public Key Infrastructure (PKI) specifies roles for nodes to take up in order to manage keys and their certificates [25]. Centralization causes bottlenecks in the system and is to be avoided in IoT settings which are flexible by nature [6]. Web of Trust is a type of self-managed PKI where no recognized authorities are present for regulation [49]. Users decide for themselves what nodes to trust as *signers* and this decentralized approach maps well on dynamic networks.

4.2.1 Pretty Good Privacy

Web of Trust's most popular application is the Pretty Good Privacy software [42]. PGP relies on trust relations defined at pre-deployment time to indicate what peers are deemed reliable to sign. *Signing* is the process of creating a new certificate and the performer is called a *signer*. Only one intermediate signer between two parties is allowed because of the lack of trust propagation mechanisms. A party trusts the direct signers but not a signer-of-signer.

Certificate Format

A certificate is a signed document binding a public key to an owner. If signature verification proved successful and the signer in question is trusted, a node accepts the associated key. Certificate Authorities make use of the X.509 format for signing [41] but PGP being fully decentralized has its own format. Its contents are condensed to the following fields:

- **Owner Identity** an identifier for the owner on the network.
- **Owner Public Key** the public key which is bound to the certificate as a proof of ownership.
- **Signer Identity** identifier of the signer. Participants performing a verification need to possess the signer's key which is only possible if a trust relation already exists.
- **Signature** its input are the previous three fields. A node having the signer's public key verifies the signature and decides whether to accept based on the verification.

Parameters

Pretty Good Privacy has local parameters custom to each node and some global parameters applicable as a whole. We will see in future how ClaviChain has the same parameters plus an additional one for chain length. Participants indicate which nodes are deemed reliable to be signers at pre-deployment time. If no specification were to be provided, it is implied that no trust exists. Certificates are easier to share when more nodes are trusted.

4.2.2 Propagation Model

Pretty Good Privacy involves at most 3 participants in the transaction of certificates: an owner, the signer and the recipient of the certificate. The actions of signing and displaying a certificate are triggered by the requesting node. A certificate reaches the node if a direct signer is available.

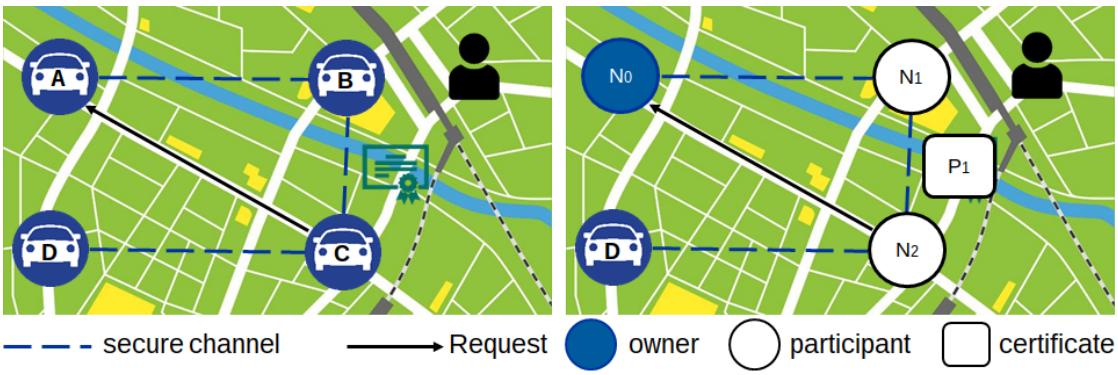


Figure 4.1: Chapter 2 introduced an Uber-like taxi service as a use case to show he issues of PGP applied in IoT settings. 2 architectures were proposed, we now cover the decentralized version. Cars A, B, C have linear trust relations. A and D are unable to obtain each others' certificates as no direct signer is present. Car N_1 signs the certificate P_1 which is fetched by car N_2.

After receiving a request, the owner of a key *pushes* a request to its trusted neighbour to sign a certificate binding its key. The original requester perceives the new document at one of its trusted neighbours and *pulls* it for further inspection.

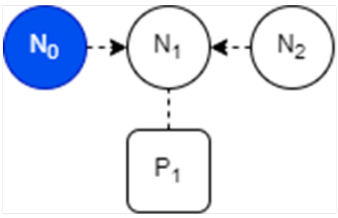


Figure 4.2: Propagation follows a push-pull mechanism between the owner and requester. N_0 in blue pushes a request to N_1 to sign a certificate. It obeys and creates P_1 which N_2 will detect and pull for verification. The certificate is accepted after verifying its signature.

4.2.3 Conclusion

WoT-based trust models have no centralized signers and are thus more flexible. Any participant is able to create digital certificates. Systems are also more fragile because of the lack of supervisors with certificate revocation a problematic task at hand [15]. PGP software requires users to identify trusted participants for certificate propagation and only allows direct trust between shared intermediates. Passing certificates over multiple nodes[2] forming intermediate chains is not possible which limits PGP in its propagation of keys. Relaxing the constraint by allowing for intermediate chains in the network improves propagation but increases the risk of forgery. No verification mechanisms are in place to verify a sequence of signers. A malicious participant could easily introduce a fake signature in between and compromise the network.

4.3 Web of Chains

Provenance concerns the study of the origin, creation and propagation of data [10]. Section 3.4 of related work gave a summary on existing provenance mechanisms. As to allow for multiple *signers* between a key owner and recipient, it was decided to integrate hash fields and pointers to link certificates together. The public key of the previous *signer* is contained within the current certificate which makes for a public-key linked chain. Multiple intermediates cooperate to pass a certificate around and the pointers to each other allows a receiver to control all intermediates and the documents that they store. Verification is not limited anymore to one document put on the assessment of a linked sequence. Web of Chains is the name of the model using this new certificate format: it is a decentralized trust model similar to Web of Trust. Chains of *signers* are possible and all members display a public certificate which doubles as a provenance record.

4.3.1 ClaviChain

ClaviChain is the real-life implementation of proposed Web of Chains model whereas PGP implements the Web of Trust. Its name is derived from Latin *clavis* or *key* alluding to the use of public key linked chains. Verification is iterative and depends on the chain length or the number of involved *signers*. A certificate is verified upon which the public key of previous *signer* is recuperated from its field and used to verify the previous certificate in the previous *signer*. Fetching and verifying documents of the chain is iterative and repeated until the owner is reached. A correct chain is suggested if no inconsistencies emerge.

Certificate Format

A format in PGP contains an identifier for the owner and the signer along with a public key. ClaviChain also has these fields and additional ones: pointers to the issuer, a cumulative hash and the issuer's public key. The format's information payload is grouped in three components:

- **Source Description** information on the previous *signer* in the chain. Its identifier and public key is contained as well as a hash of previous certificate. The hash is used to check whether the current and previous certificate are consistent. A signature is used to verify the fields against tampering.
- **Chain Condition** a field to indicate a chain's status. If some previous verification suggests the chain has been tampered with, the current field is used to indicate this. Having such field saves future parties from controlling the whole chain.
- **Meta Data** useful system information such as the list of successors in a chain or previous performance of (ex-)chain members. This field was added in evaluation and meta-data is stored and retrieved here for further processing.

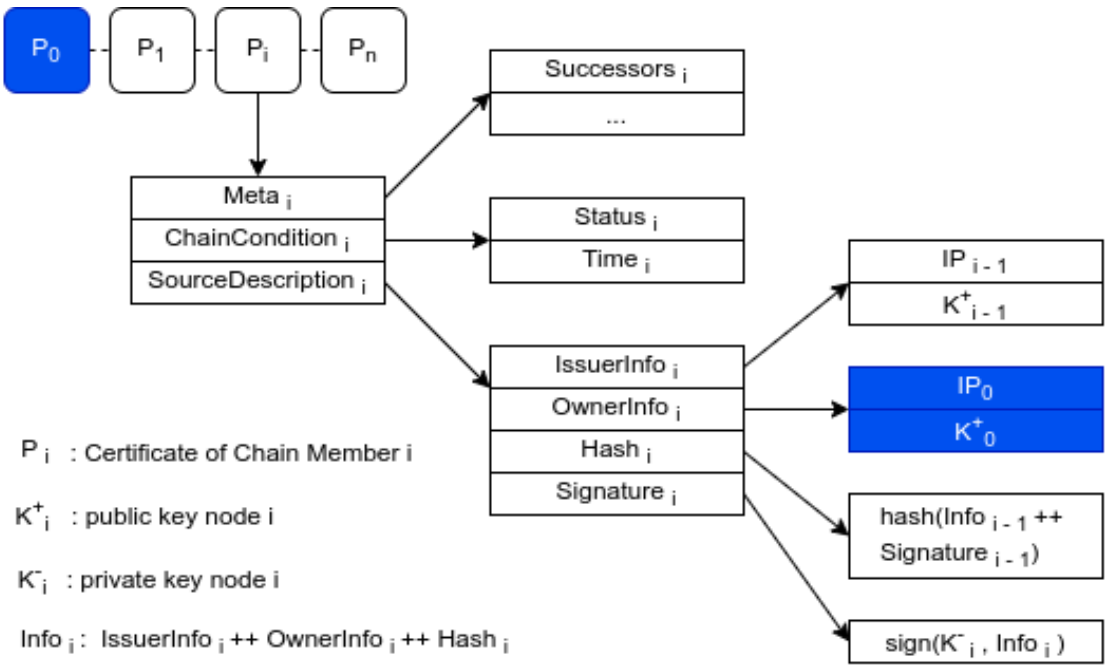


Figure 4.3: A certificate is composed of meta, chain condition and source description fields. The memory overhead is quite higher than in PGP but allows for better key propagation. Blue cases indicate the owner key and its identifier which is the essential information of the chain.

Notice how the source description contains the actual signature, owner key and identifier. Other fields are related to chain consistency and meta-data and don't need to be verified.

Chain Length as Global Parameter

Section 4.2 described the global parameters as configurations on how much certificates are to be collected in order to accept a key. All participants abide to global parameters whereas the local parameters are custom to an individual node and indicate trust relations. ClaviChain has chain length as additional global parameter as it indicates the maximal length for provenance chains. An upper length means some upper bound resource usage is computable and it prevents forgers from spawning infinite chains with collaborating nodes.

4.3.2 Signing Certificates

Creating certificates or the act of *signing* is done by any participant in the Web of Chains as it is completely decentralized. A *signer* is a party creating certificates. The input for a new certificate is a public key, the owner's identity and optionally a previous certificate. If a certificate extends a previous chain the last certificate needs to be given. A certificate or *root* indicating the start of a chain does not extend an existing certificate.

ClaviChain signs two types of certificates:

- **Root Certificate** which has no previous as it is the origin of current linked chain. Such document is displayed by a key's owner and indicates the *root* of a chain. If the iterative verification process has reached the root without encountering inconsistencies, then the corresponding key is valid. Signing a root in ClaviChain is similar to signing in PGP.
- **Intermediate Certificate** are documents that are generated from some previous certificate, extending its chain. It points to a previous certificate eventually ending in a *root* certificate. An existing document is required to make a new one and the chain is *extended*.

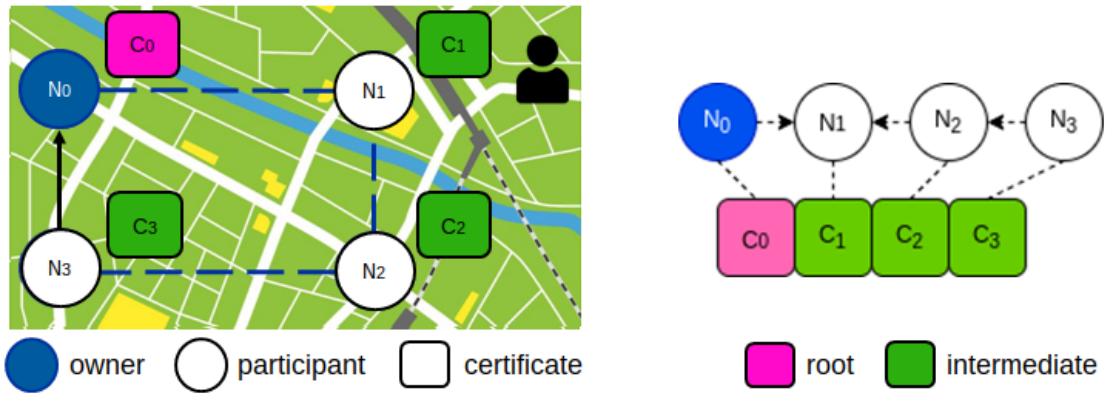


Figure 4.4: Pink boxes are root certificates to indicate the origin of a chain. Green boxes are intermediate certificates and a sequence of these ends in a root.

4.3.3 Verifying Certificates

Controlling a certificate against tampering includes linked chain members as opposed to PGP which only includes the current document. The complexity of verifying a linked chain depends on its length as more members mean more certificates to consult. An upper bound for complexity is the global chain length parameter as any chain exceeding the limit is invalid.

Verification is an iterative process and each iteration verifies the current signature with the public key of current intermediate. The party which performs the verification needs to possess the first intermediate's public key, all other keys are accessed from a certificate as a chain is linked with public keys. *DeepCheck* refers to the algorithm performing the iterative verification.

Algorithm 1: DeepCheck

```
1:  procedure VERIFY(certificate, public )
2:      current ← certificate
3:      length ← maximal chain length parameter
4:      lengthleft ← length
5:      goto loop.
6:  top:
7:      previous ← issuer of current
8:      owner ← owner of public key
9:      if previous = owner then
10:         correct ← return for a correct verification
11:         return correct
12:     else
13:         current ← certificate of previous
14: loop:
15:     verification ← Verify signature of current
16:     if verification = false and length = lengthleft then
17:         currenterror ← return for a mistake in the first issuer
18:         return currenterror
19:     else if verification = false  then
20:         previouserror ← return for a mistake somewhere in previous chain
21:         return previouserror
22:     else
23:         goto top.
```

Possible Return Values

The pseudo code has a return value for a successful verification and failure when some inconsistency is detected. It is possible to pinpoint where in a chain an inconsistency occurs by controlling the member documents separately.

4.3.4 Propagation Model

We now explain the course of actions taken in propagating certificates with the Web of Chains model. The taxi service use case from Chapter 2 provides an illustration. Drivers N_0 in blue and N_3 in white are in a predicament of not having any direct issuers to share keys.

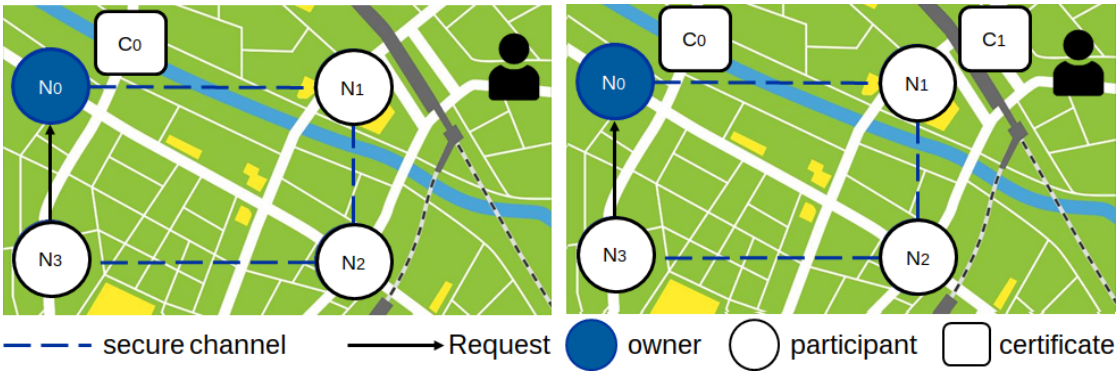


Figure 4.5: Owner node N_0 has its own root certificate C_0. The chain is expanded by signer N_1 which first verifies the certificate before verifying its own.

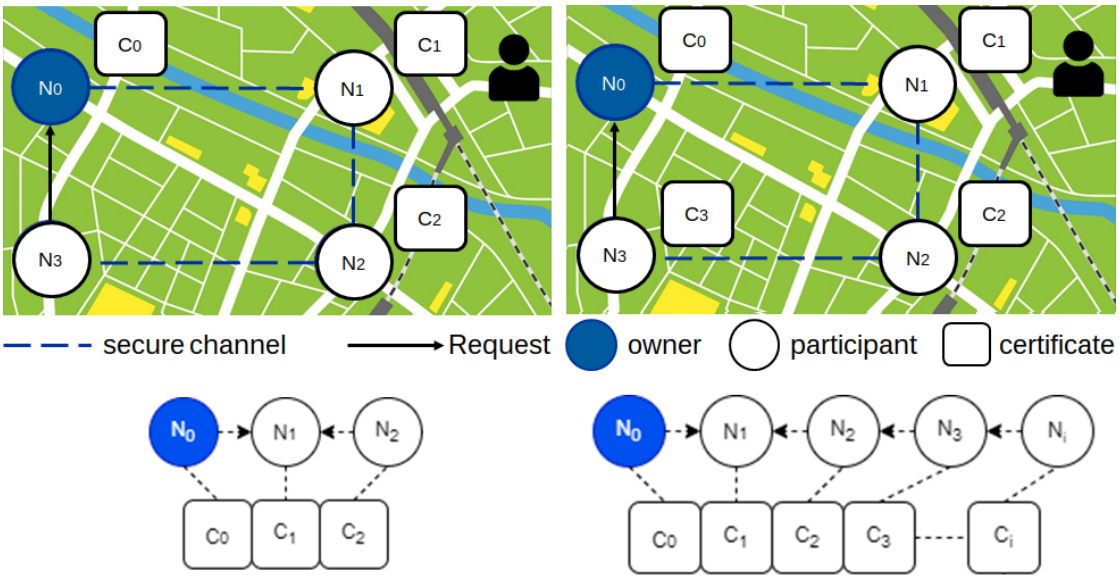


Figure 4.6: N_2 will pull and verify certificate C_1 and generate certificate C_2. N_3 will in its turn verify the whole chain (N_2 – N_1 – N_0) and generate its own certificate for extension. N_3 will have more trouble verifying the chain than the previous signers as the chain keeps becoming longer and longer meaning propagation is slower and slower.

Verification of all chain members happens before signing a new certificate. After the chain is consistent, the recipient makes a intermediate certificate in order to *extend* the chain. An additional member is now part of the sequence. Propagation is slower for longer chains due to added overhead of having more certificates to verify.

4.3.5 Conclusion

Web of Chains has added fields in its format to link certificates together into a public-key linked chains. Verification is no longer constant as multiple documents are added in the process. Assessing correctness in this approach yields better results as a whole chain is verified. Performance is much steeper as a downside which was to be expected when making a model more robust [8]. An advantage is improved propagation as multiple intermediates are now possible.

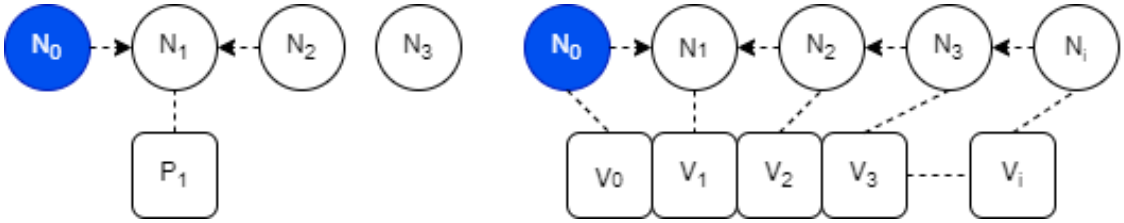


Figure 4.7: Nodes *push* a request to sign and display a certificate binding their key onto a trusted node. Participants wanting their keys *pull* the displayed documents in order to verify them. Nodes in ClaviChain add a new certificate to their collections for others to *pull* them in.

4.4 Discussion

Our research efforts considered a range of policies for the Web of Chains to deal with inconsistencies in a linked chain of *signers*. Unreliable participants would be identified and replaced. A whole array of policies were developed and some initial tests were performed with delayed malicious participants: forgers who only falsify displayed certificates after some time has passed. Interesting pointers for future work but eventually not pursued in this dissertation.

4.5 Metrics

Allowing for multiple intermediate *signers* with proposed format is bound to affect the properties of distributed systems. Metrics are used to examine aspects of a system abiding to some model. Chapter 6 contains the evaluation of the proposed models and uses the metrics of this section in order to study their advantages and disadvantages.

Smart Environments
A device is *smart* if it obtains and applies knowledge independently [5] and *smart environments* are comprised of such devices. Evaluation in Chapter 6 is performed on a simulation which mimics *smart environments*. Metrics are chosen with such target systems in mind.

4.5.1 Effectiveness

The first aspect to measure is described as the extent in which a trust model is successful in reaching design objectives [50]. Objectives in this dissertation are to improve key propagation and correctness of certificates. Forgery is more accurately detected in the Web of Chains and keys are propagated even further with the added provenance in format. Its trust architecture remains decentralized. Three metrics are identified to measure effectiveness:

- **Coverage** is the extent in which nodes are able to do predictions on certificates in terms of authenticity [18]. Unforged certificates are deemed authentic. For a scenario where all nodes have direct trust relations to each other, coverage as a metric is maximal or 100 percent. Nodes are all adjacent to each other. The formula below gives coverage:

$$C = \frac{\sum_{j=1}^{M} \frac{N_j}{N}}{M}$$

  M is the total number of network participants and N_j the number of certificates predictable to user j whereas N is the total number of certificates. C represents the obtained coverage which in the ideal case –when all nodes possess the public key of their peers– equals 1.

- **Accuracy** is the extent in which correct assessments on authenticity are made. It is the average fault nodes commit when determining authenticity. A perfect accuracy metric of zero percent indicates forgery is detected with no faults. The formula below gives accuracy:

$$A = \frac{\sum_{j=1}^{M} \left[\frac{1}{N_j} \sum_{i=1}^{N_j} \left| r_j(t_i) - g(t_i) \right| \right]}{M}$$

  M is the total number of participants whereas N is the total number of certificates with N_j the number of predictable certificates for a user j. t_i represents a certificate and $r_j(t_i)$ its perceived authenticity whereas $g(t_i)$ is the ground truth of its authenticity.

- **Spread** is a more quantitative depiction of effectiveness and measures the amount of collected keys per individual node relative to the total number of keys. It indicates to what extent secure channels were opened on a distributed network. The novel format discussed in this chapter is expected to improve the metric. The formula below gives spread:

$$C = \frac{\sum_{j=1}^{M} \frac{N_j}{M-1}}{M}$$

  M is the total number of network participants and N_j the number of collected keys of participant j. By dividing it with $M-1$, we get the percentage of the participants with which a secure communication channel is opened. We take the average spread of all participants in order to compute the final metric.

4.5.2 Efficiency

The second aspect measures the amount of resources consumed for completing a fixed task [50]. Provenance in the certificates makes for a significantly higher memory usage and runtime performance for operations. Evaluation consists of benchmarking the model implementations performed on only one virtual machine.

Completion Time and Memory Usage

Algorithms distinct to each model are measured on resource consumption: completion time and memory usage from which complexity is estimated. Benchmarking is the technique of repeatedly running an algorithm in order to determine these runtime properties. The simulation relies on external packages native to the Elixir code base which we briefly mention in Chapter 5.

4.6 Conclusion

Web of Chains links certificates together to a public-key linked chain by using the format proposed in section 4.3.1. The new trust model is essentially Web of Trust using this format. Its implementation is referred to as ClaviChain. Our goals are to allow for multiple *signers* between two parties as to improve key propagation. The metrics of section 4.5 help us to examine the model more closely when it comes to key propagation and authenticity of certificates. A simulation customizable on various parameters was developed to validate our claims. It was written in Elixir which is a programming language built for the development of distributed applications [37]. Its code base maps well on the properties of *smart* environments.

5 Implementation

This chapter discusses the implementation of our simulation in a top-down approach. A starting point for its development was the Potato middleware [14] written by advisor Christophe De Troyer(PhD candidate SOFT, Vrije Universiteit Brussel). PGP and ClaviChain are modelled as groups of communicating virtual nodes and the Elixir code base was chosen for its crash-resistant nature. It uses the actor model [37] which maps well on the Internet-of-Things and its properties. Code segments are grouped in *modules* used to initialize processes on virtual machines. A node is expected to start a couple of processes with the written modules in order to behave in an expected manner. Model-specific behaviour is abstracted away in separate modules and are selected with command-line arguments. Virtual machines are customizable on their trust model, honesty and trust relations with other nodes.

5.1 Architecture

Decentralized trust models allow any participant to create certificates: it is the act of *signing* with performers called *signers*. Chapter 5 discussed PGP and ClaviChain (abbreviated *CC*) as decentralized approaches to trust with certificates shared to open secure channels. A novel certificate format using multiple *signers* is used by ClaviChain in order to improve key propagation.

Abstractions were written in such a way that their custom behaviour is isolated in the *CC.ex* and *PGP.ex* modules. Four methods implement a model's behaviour and there is a tight coupling with the *entry_point.ex* module which serves as an entry point for the whole network. Methods decide what needs to be done with requests received at the entry point.

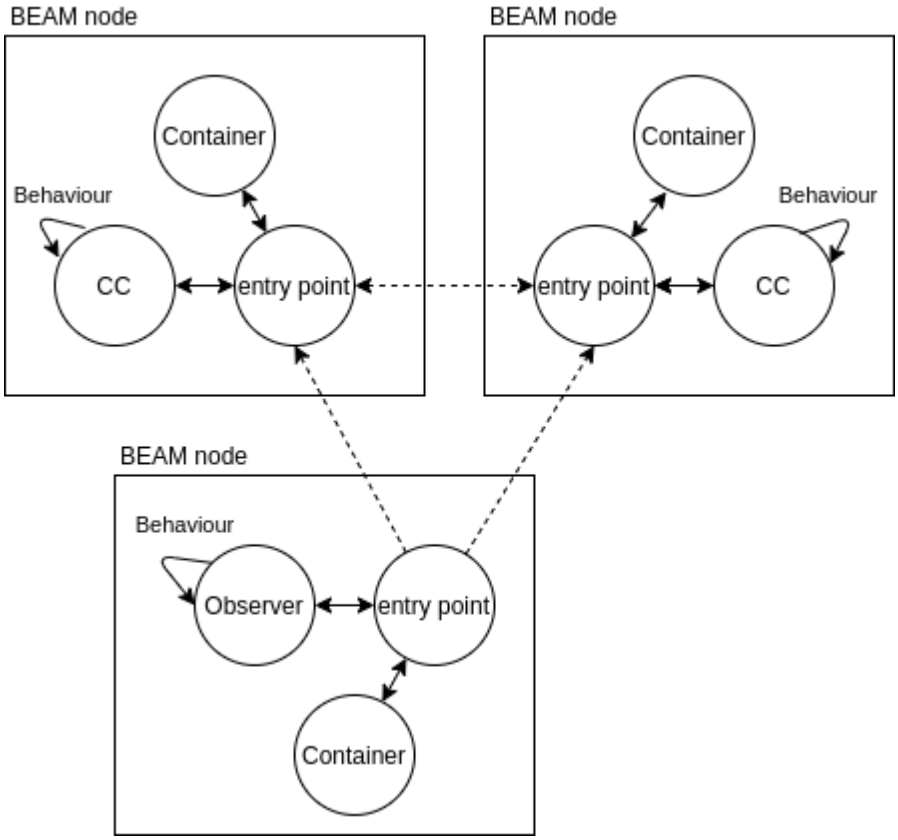


Figure 5.1: A BEAM node starts a behavioral module tightly coupled to the entry point. Observers are passive and compute metrics for evaluation. A triplet of nodes is shown initialised with ClaviChain behaviour: some process (circle) is initialized with behaviour module *CC.ex*. A process initialized with module *entry_point.ex* receives all network packets.

Behavioral modules are closely coupled with the entry point and implement specialized behaviour. A BEAM node either actively tries to open secure channels or passively samples data for examination. It depends on one of the following modules:

- **CC.ex** implements ClaviChain behaviour following the Web of Chains model.
- **PGP.ex** implements Pretty Good Privacy behaviour abiding to Web of Trust.
- **Observer.ex** implements passive behaviour where data of participants is sampled in order to compute the coverage, spread and accuracy metrics discussed in section 4.5.

The selection of one of these modules in order to display some custom behaviour depends on arguments inserted at the command line.

5.1.1 Propagation

entry_point.ex is initialized for every node and implements general behaviour for certificate propagation and communication. The entry point module is tightly coupled with other processes on the virtual machine accounting for specialized behaviour.

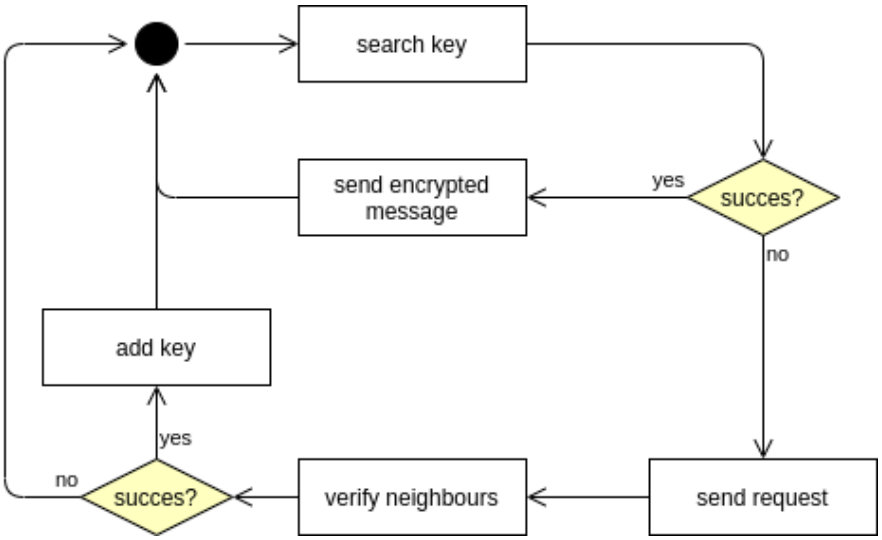


Figure 5.2: state diagram showing the set of actions taken by a participant to broadcast its message to peers. Requests are only pushed for missing keys.

Opening secure channels in the simulation follows these four steps:

1. **Send message or request key** a virtual machine broadcasts encrypted messages from the entry point to all known nodes in the network. If a key is missing, a message is sent to corresponding node requesting the missing key.

2. **Push sign-request to trusted** upon receiving a request for its key, the owner reacts by sending a request to its trusted neighbourhood to sign a certificate binding its key.

3. **Sign and display** upon receiving a sign request, the trusted node spreads the key by signing a certificate and displaying it in its collection. The way how signing is performed depends on the model and is specialized by one of the behavioural modules.

4. **Verify neighbours** the entry point has a default loop which is used to verify its trusted issuers for new certificates. A node computes the missing certificates and verifies its neighbours for the missing documents. Receivers then check whether they are in possession of such certificates and if so send them back. A node accumulates such certificates until it has enough to accept. Whether a certificate is calculated to display depends on used model.

Notice how the process takes 2 asynchronous *casts* instead of 1 synchronous *call* to open a channel. A reason why asynchronous requests are used is that synchronous requests are very prone to network time-outs when used for much-repeated operations. It is not reliable in practice.

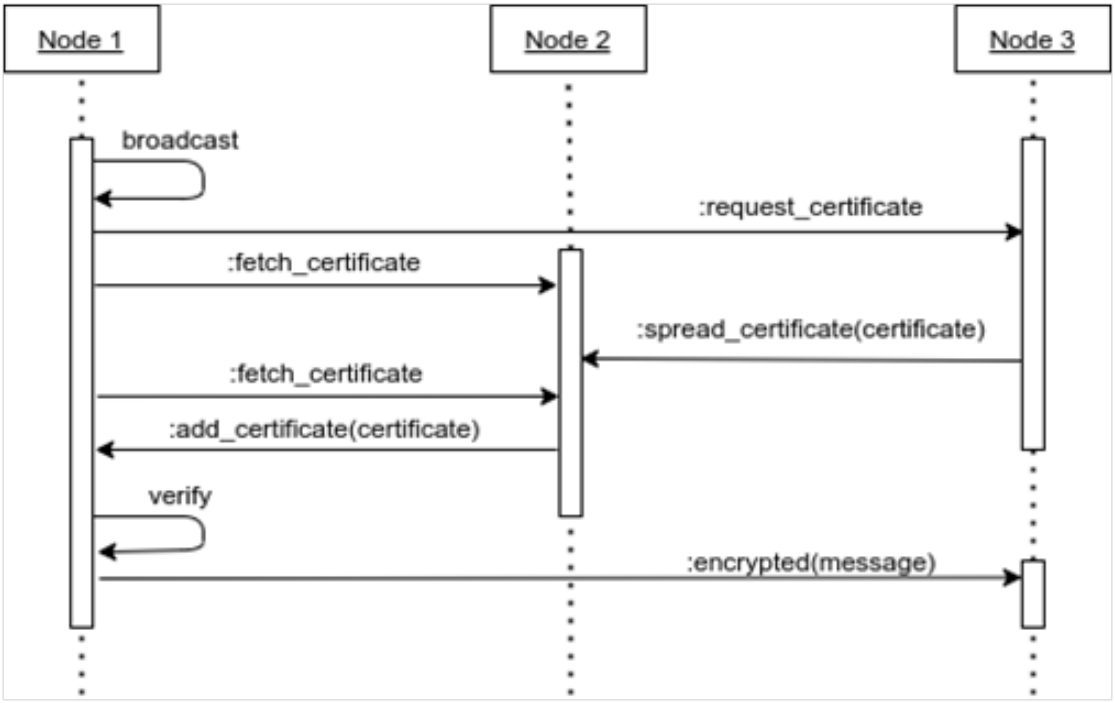


Figure 5.3: Sequence Diagram describing the necessary steps to open a secure channel. What is not shown is how adding a certificate in ClaviChain extends the current chain of *signers*.

5.1.2 Roles

Command-line arguments determine the behaviour of a virtual node. We classify a virtual machine as either a participant actively managing and receiving certificates or as an observer who collects data for examination. Behaviour of network participants are to broadcast encrypted messages to peers in the network. Keys are requested when they are missing.

```
def handle_info(:behaviour, state) do
    time_interval = 5000 # 5 seconds
    message = "what's up from: " <> Kernel.inspect(Node.self())
    Observables.Entry.Point.broadcast(message)

    Process.send_after(self(), :behaviour, time_interval)
    {:noreply, state}
end
```

Listing 5.1: A process uses the *GenServer* behaviour module to implement generic server behavior. Broadcasting messages is done in a loop in order to initiate communications.

5.2 Pretty Good Privacy

A participant node is initialised with *PGP.ex* in order to mimic Pretty Good Privacy which abides to the Web of Trust model. Custom implementations are provided in order *sign*, *verify* and *add* certificates. The *miss* method computes the missing certificates in the collection. Signing a new certificate in PGP requires the key of the owner and its identifier as well as that of the signer. Verification in PGP has no real surprises and its source code is not that interesting to discuss in detail. It starts by hashing the certificate's content then comparing the signature with the issuer's key. *Add* and *miss* operations are used to add certificates to a node's collection and to determine the missing certificates. *add* in ClaviChain has a few lines of code more to generate a new certificate which makes for quite a resource overhead in evaluation.

5.3 ClaviChain

The module *CC.ex* contains all source code for ClaviChain's custom behaviour. A virtual machine initializing a process with this module displays ClaviChain behaviour and abides to the Web of Chains model. Operations in this module are more complex because of the new fields for provenance in the corresponding format: pointers to issuers, a cumulative hash, etc. All the new fields are included in the signature of the certificate.

5.3.1 Signing Certificates

Making new certificates in ClaviChain demands significantly more resources with the inclusion of back pointers, hashes and additional keys. Section 4.3.2 explained how a distinction is made between certificates indicating the origin of a chain and certificates which are pointing to a previous document. The former is called a *root* and the latter an *intermediate*. Generating a signature starts by hashing the certificate information into one string used as input for the certificate's signature. A root has no previous certificate and includes nothing when hashing or *nil* in Elixir terms. All information and the signature have a field in the document.

```
1  def sign(certificate, issuer_pid, issuer_pub, signer_public,
       signer_private) do
2    generate(certificate, issuer_pid, issuer_pub, signer_private)
3  end
```

Listing 5.2: Signing can either happen for a root certificate –in which case the certificate parameter is nil or for a previous certificate in which case the its core information is hashed to serve as input for the new signature.

5.3.2 Verifying Certificates

ClaviChain performs verification with the iterative *DeepCheck* algorithm discussed in section 4.3.3. Its upper bound performance is determined by chain length as any chain exceeding that length is invalid. A new certificate belonging to the chain is fetched and verified in each iteration. The back pointer of current certificate is used for the next iteration and the process repeats until the root of the chain is reached. It is the stop condition of the algorithm. The chain and its initial certificate are correct if no inconsistencies are detected.

```
1  def verify(certificate, public) do
2    try do
3      # See what the maximal chain-length is.
4      {chainLength, rest} = Integer.parse(System.get_env("ChainLength")) ||
           {3, ""}
5      verification_result = verify_length_iter(certificate, chainLength,
           public)
6      {issuer_pid, issuer_public} = issuer_info(certificate)
7
8      # Give back result of the verification.
9      verification_result
10   catch
11     # Some time-out. Try again.
12     e -> verify(certificate, public)
13   end
14 end
```

Listing 5.3: A significant issue with the fetching process were the frequent network time-outs due to synchronous requests which are time critical. The request is made in the body of *verify_length_iter*.

Synchronous Version

A *call* is a synchronous request –a reply is expected within reasonable amount of time. An asynchronous request where no reply is expected is named a *cast*. Synchronous requests make the algorithm prone to connection time-outs in a real-life set-up. So much so that the *container.ex* module was added for the frequency of crashes.

```
1      try do
2        # Previous certificate.
3        GenServer.call({Observables.Entry.Point, issuer_pid}, {:
             give_certificate, owner_pid})
4
5      catch
6        e -> fetch_sync(issuer_pid, owner_pid)
7      end
8  end
```

Listing 5.4: A time-critical *call* is performed in *verify_length_iter* which performs a request to the previous issuer for its stored certificate. A certificate is a record in the chain. All of them are fetched iteratively up until the root.

Using synchronous calls was acceptable for a set-up with 10 or so nodes but unpractical for larger systems which is why a version with asynchronous *casts* was implemented as an alternative. A possible use of the synchronous version is in benchmarking. It is fragile and fast: perfect to assess resource consumption in the *DeepCheck* algorithm.

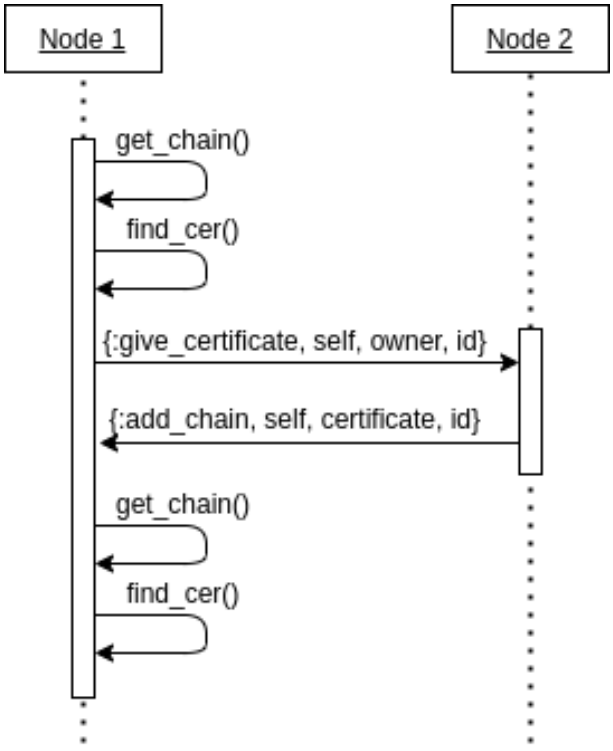
Asynchronous Version

DeepCheck with asynchronous verifications is used to examine groups of virtual machines. Propagation is slower with asynchronous requests and is difficult to benchmark due to inconsistent latency. Time-outs seldom appear which makes the algorithm more reliable but also slower with asynchronous calls. It receives the wanted certificates by checking its mailbox. The certificates that were sent were answers –not replies– of the nodes which possess the wanted certificates.

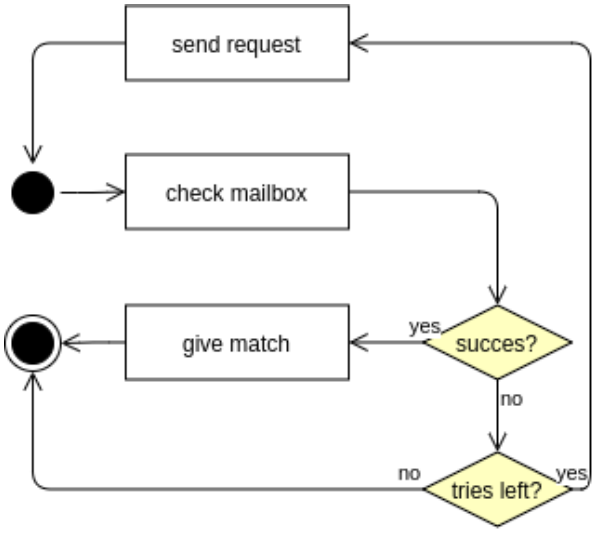
```
1  def fetch_async(cer, issuer_pid, owner_pid) do
2      tries = 50
3      fetch_previous( issuer_pid, owner_pid, id(cer), tries ) #, identity,
           tries)
4  end
5
6  # Asynchronous version.
7  def fetch_previous(issuer_pid, owner_pid, id, tries) do
8      # You are at the root.
9      chain_cer = Network.Model.CP.get_chain()
10     match = find_cer(chain_cer, owner_pid, issuer_pid, id)
11
12     cond do
13      match == nil && tries > 0 -> GenServer.cast({Observables.Entry.Point,
            issuer_pid}, {:give_certificate, Node.self(), owner_pid, id})
14                                   fetch_previous(issuer_pid, owner_pid, id
                                         , tries - 1)
15      match == nil -> nil
16      true -> match
17     end
18 end
```

Listing 5.5: This function is called in *verify_ length_ iter* and will do an asynchronous request to the previous issuer or member of the chain. Notice how *Node.self()* is provided in the message itself in order for the recipient to know where to send the answer to. This really shows how asynchronous the process is: two messages are sent to each other instead of just one.

Evaluation uses the synchronous version for benchmarking –for more concise measurements leaving out latency– and the asynchronous version for examining groups of virtual machines and their interaction. The sequence diagram for asynchronous fetching is given below.



(a) Sequence Diagram for asynchronous fetch.



(b) State Diagram for asynchronous fetch.

Figure 5.4: Diagrams for asynchronous fetch.

Return Codes

Verification in ClaviChain does not give back a boolean with only 2 possible values but a numerical value. It is a code that signals success or errors with their type. Negative values indicate a kind of failure whereas positive values indicate a successful verification.

5.4 Observer

One node is started which does not interact with its peers and thus does not propagate certificates to open channels. It merely collects information of the participants in terms of performed calls, certificates and keys owned, etc. The collected information is used to compute performance metrics as coverage, spread and accuracy of which it contains all code to do so.

5.4.1 Information Gathering

Collecting node information with synchronous calls caused too much time-outs for accurate measurements to take place. The *observer.ex* module has a loop which sends asynchronous *:update* calls to the entry point of other participants which promptly send an answer with fresh information back. The sender puts its network identity in the message for the recipient to know where to send its answer to. It is a rudimentary but practical solution.

Listing 5.6: This loop sends asynchronous update requests to the participants which promptly send updates back asynchronously for the process to store.

```
1  def handle_info(:update, state) do
2
3     Enum.map(Node.list(), fn pid -> get_update(pid) end)
4     # Don't forget to call next loop.
5     Process.send_after(self(), :update, 1000) # Every second.
6     {:noreply, state}
7
8   end
```

5.4.2 Metric Computation

Metrics are measurements which represent some qualitative property of the system. We identified coverage, accuracy and spread as the main metrics for evaluation in section 4.5. They allow for a more precise comparison to be made between the models. ClaviChain computes coverage by taking all accepted certificates of a node and counting all other certificates that are part of their chains. Each certificate has an identifier of the chain to which it belongs to ease coverage calculation. Accuracy is computed by making the node perform an assessment of a certificate's reliability. The ground truth indicates the correctness of a certificate and is measured against perceived correctness. We get the average fault for network participants. Showing all code segments here is not very interesting as they are literal translations of the formulas proposed in Approach section 4.5.

Listing 5.7: Code segment for the Mean Absolute Error (MAE).

```
10
11     def median_absolute_error(all_certificates, collection) do
12      # users
13      users = Node.list()
14      # No. of users
15      no_users = length users
16      # All certificates (distinction between providers).
17      all_absolute_errors = Enum.map(users, fn pid ->    absolute_error(pid,
             all_certificates, collection) end )
18      mae = Enum.sum(all_absolute_errors) / no_users
19
20      mae
21    end
```

5.4.3 Packages

The *Benchee*[1] library was used to analyse the resource consumption of methods. *Benchee* allows to specify what methods are to be executed, for how long and to which csv file the results are to be written. Specifying how long a method is to be executed makes it difficult to estimate the precise number of repetitions. It was a process of trial and error with tests repeated in case the required 30 executions were not met. *Benchee* implements the csv I/O with its plugins[2].

5.5 Conclusion

This chapter discussed the code modules implemented in this dissertation and the communication flow between them. We then discussed main *roles* identified in the simulation and how the modules implement the behaviour of trust models. ClaviChain's implementation was explained with a focus on the *DeepCheck* algorithm for the verification of certificates. Abstracting away model-specific source code from more generic code proved be a significant challenge in development. It had to be put inside separate *modules* customizable at the command line. Synchronous requests proved to be a mayor difficulty in the simulation's development. Synchronous calls are time-critical and fast but fragile as they caused a significant occurrence of network time-outs. It was deemed a better choice to go for asynchronous requests: sacrificing speed for reliability.

[1] https://github.com/bencheeorg/benchee
[2] https://hex.pm/packages/csv

6 Evaluation

In this chapter, we evaluate the ClaviChain (CC) implementation against Pretty Good Privacy (PGP). Models are recreated in the simulation and experiments focus on the properties of individual machines and collective groups. Four methods implement model-specific behaviour and they are repeatedly executed in order to estimate a model's resource needs. Another part of evaluation mimics real-life smart environments with groups of machines broadcasting and sharing information independently. Virtual machines are customized on intent or whether they open correct channels or perform harmful acts of forging by sharing incorrect keys. Optimistic experiments focus on the sharing of keys and more realistic ones on robustness or whether keys are not tampered with. Our goal in this chapter is to determine the resource requirements of individual machines followed by key propagation and robustness in groups of machines. We expect key propagation to improve with linked chains to the detriment of resource consumption.

6.1 Methodology

Experiments are performed on individual machines and on cooperating groups as well. On one hand, the custom implementations of models are measured and on the other, properties of distributed systems. They are initialised with ClaviChain or Pretty Good Privacy behaviour with Evaluation following 3 techniques:

- **Benchmarking** the repeated execution of model-specific methods to determine resource consumption: average completion time and memory usage.

  Models differ in terms of four interface methods: signing, verifying, adding certificates and computing missing documents. Each of them is benchmarked at least 30 times.

- **Profiling** after measurements on custom methods limited to a sole virtual machine, the evaluation measures runtime properties of systems initialized with the models. Profiling assumes an ideal case as it does not include the influence of forgers on performance.
  The following aspects are studied:

  - The effect of network size on certificate propagation. Experiments give indications whether ClaviChain scales well for bigger networks.
  - Effects of chain length on propagation. Longer chains imply more intermediate signers and thus more propagated certificates. The amount of participants is not changed.

- **Monitoring** this part considers the influence of forgers and makes for a more realistic setting. Linking certificates is shown to help in the detection of forgery even when the rate of forgers increases.
  The following aspects are studied:

  - Rate of forgers and its effect of on correctness on a network of 50 virtual machines. It is done to show the influence of linked chains on the detection of forgers.
  - The effect of alternative *signers* on key propagation. Provenance improves the detection of forgers which means the reliable alternative certificates are used as replacement.

6.1.1 Metrics

Certificates and their keys are linked together into chains and this allows for multiple parties between a key owner and a recipient. Quantitative measurements on the impact of multiple intermediates are the number of certificates on the network. The metrics discussed in section 4.5 are used to obtain more qualitative results. We give their description here:

- **Coverage** is the degree in which an average node can do predictions on the certificates in circulation. Some existing knowledge about the node needs to exist in order to make this possible: a trust relation, the a correct chain, etc. High coverage implies lots of available certificates from various signers.
- **Spread** is the extent in which keys reach the participants in a network. Including the metric was necessary as ClaviChain signs more documents for the same coverage in PGP.
- **Accuracy** represents the amount of correct predictions on certificates. Accuracy shows how robust a trust model is in highly malicious networks by consistently making correct predictions.
- **Completion Time and Memory Usage** when put together represent the resource consumption of a code snippet. Measurements are obtained by repeatedly running some method, at least 30 times in our experiments.

ClaviChain has multiple signers between two parties exchanging keys. Each intermediate stores a certificate which is linked together into a coherent chain. Coverage, spread and accuracy are improved in this approach for increased resource consumption. Forgery by signers is detected even better and keys are propagated even further. Coverage and spread are correlated and evolve to similar values when more trust relations exist.

6.2 Setup

Measurements were performed on the student's personal computer and on the *Firefly* server located at VUB, campus Etterbeek. Hardware specifications are given in featured tables. Analysis on the server was done to study the tendencies of models for increased network sizes. Different scenarios were tried in order to study their effects.

Processor	
Model	Intel Core I5-5300U
Cores	2
Threads	4
Storage	
Disk	1 TB
Main memory	8 GB
Software	
OS	Ubuntu 20.04.2 LTS
Elixir version	Elixir 1.9.1/OTP 22

Table 6.1: Personal computer hardware

Processor	
Model	Ryzen Threadripper 3990X
Cores	64
Threads	128
Storage	
Disk	2x2 TB
Main memory	128 GB
Software	
OS	Ubuntu 20.04.2 LTS
Elixir version	Elixir 1.9.1/OTP 22

Table 6.2: Firefly hardware

6.2.1 Validation

We validate our contributions based on the following points:

- Is resource consumption in ClaviChain acceptable compared to PGP?
- Is certificate propagation in ClaviChain improved compared to PGP?
- Does ClaviChain detect forgery even better with added provenance?

6.2.2 Scenarios

Scenarios represent trust relations between nodes following either ClaviChain or PGP. By experimenting with different trust distributions and the presence of forgers, one can study their effect on certificate propagation. An array of trust distributions were tested to show strengths and weaknesses of CC and PGP. See Appendix A for illustrations on their trust distributions.

Scenario 1: Simple Linear

A linear distribution does not contain any cycles and makes it easier to find certificates with a custom amount of previous nodes. Fetched certificates are then repeatedly measured on their verification process to show algorithm complexity for different chain lengths.

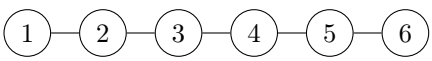


Figure 6.1: simple linear graph

Scenario 2: Cycle Graphs

A cycle or circular graph is characterized by its nodes forming a closed chain with members having an in-degree 2 [52]. Every node has exactly 2 edges connecting it with its neighbours in a cycle. Trust relations are mutual meaning both parties trust each other. Cycle graphs have an equal amount of trust relations –edges– per node and ensure equal opportunity for all participants to receive certificates. It was deemed a good choice for dynamic program analysis.

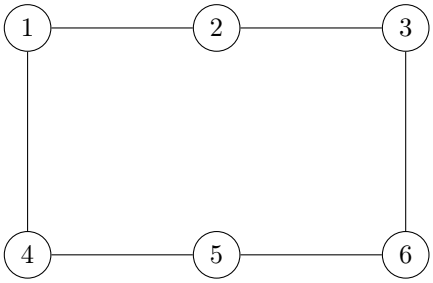
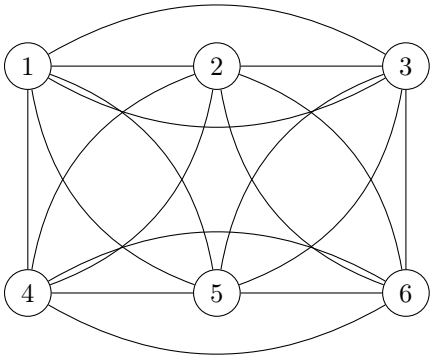


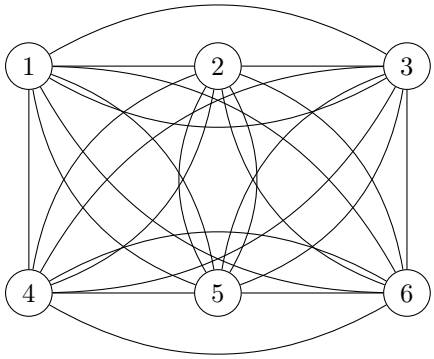
Figure 6.2: cycle graph

Scenario 3: Strongly Regular Graphs

A cycle graph is also a regular graph because all vertices have the same number of neighbours [52]. Adding more trust relations between the vertices to connect neighbours-of-neighbours makes for a strongly regular graph. Strongly regular graphs increase the number of neighbours for non-adjacent nodes (μ) and preserve its cyclic structure. Adding more neighbours makes the graph ideal to measure the effect of alternative signers on key propagation.



$\mu = 2$



$\mu = 3$

6.3 Resource Consumption

A certificate in ClaviChain has additional information on provenance such as a pointer to the *signer* node from where the chain was *extended*. Provenance is related to the document's origin. Operations on certificates are more costly because of the fields being included in the signature of the document. Benchmarking runs the methods repeatedly while measuring completion time and memory usage. Results can vary depending on available hardware [1] and experiments are therefore repeated a total of 30 times for higher confidence in results.

6.3.1 Signing Certificates

Certificates are a proof-of-ownership binding a key to its owner. ClaviChain includes a pointer to the issuer, its public key and a hash in addition to the information in existing PGP format. Signing can either include some precedent certificate in order to make an intermediate certificate or none to make a root certificate: it has no previous and is there to indicate the end of a chain.

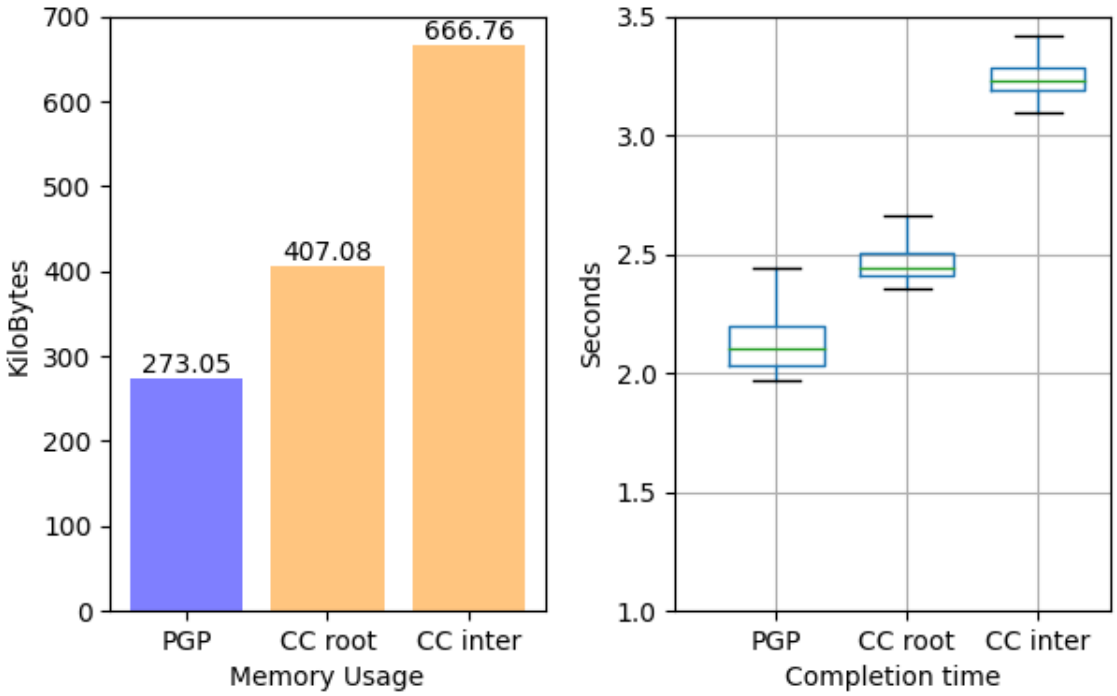


Figure 6.3: Additional fields to the format make for more than double –almost triple– the memory usage when an intermediate certificate is signed. If a root certificate has to be created, its memory usage is almost 100 KiloBytes less.

Conclusion

With added provenance, ClaviChain certificates are more costly to make. Signing an intermediate certificate from an existing one in ClaviChain is almost twice as expensive in time and memory compared to signing a root. The explanation for this is that the necessary information needs to be extracted and hashed of the input certificate to make a new intermediate.

6.3.2 Verifying Certificates

Section 5.3.2 explains how asynchronous calls make ClaviChain's *DeepCheck* algorithm very fragile in verifying certificates. Network time-outs are frequent with synchronous calls and the asynchronous version is deemed better for dynamic program analysis on groups of virtual machines. In this section, we want to measure the resource requirement of individual snippets of code. In order to cancel out network latency, the faster synchronous version is used.

In ClaviChain, the distinction is made between a root and intermediate certificate. A root refers to a certificate without provenance. It is a certificate indicating the end of the provenance chain and is stored by the owner of corresponding key. An intermediate certificate is stored by some intermediate signer and points to its previous signer or the root.

Root Certificates

Verifying a root certificate is very similar to PGP. Only one document is used in the two cases. Performance in ClaviChain is significantly higher because of the added fields in the format.

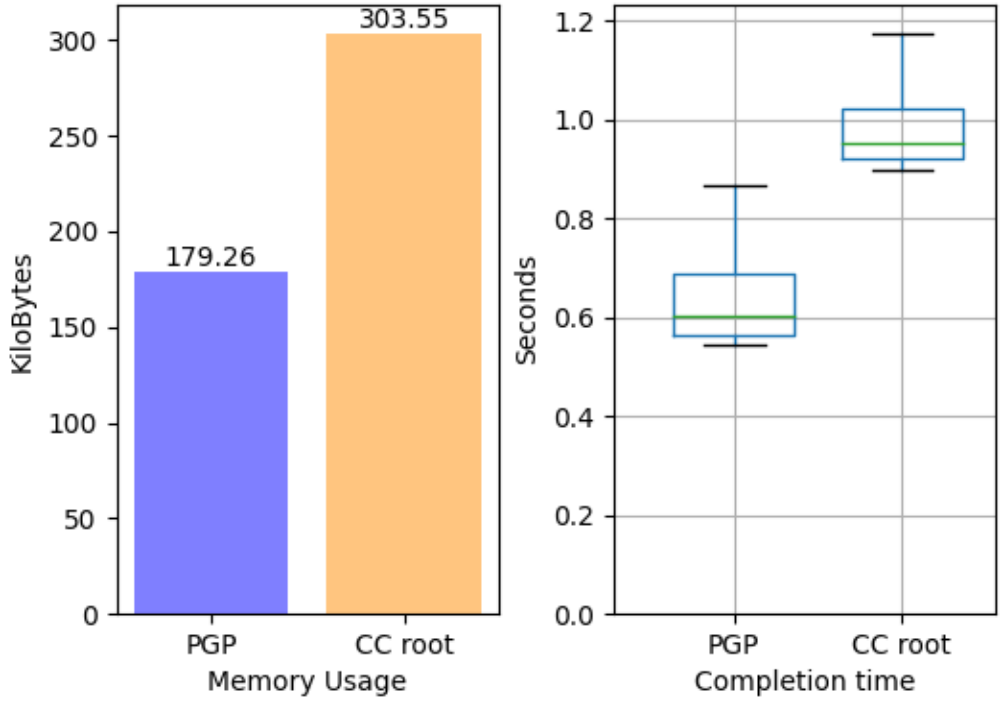


Figure 6.4: Verification properties for PGP and a root certificate of ClaviChain: in both cases only one document is considered. Verification of a root is almost double in memory usage and completion time compared to PGP. This is due to additional fields.

Intermediate Certificates

Verifying intermediate certificates has a linear performance as previous documents are included. Different chain lengths were tried in ClaviChain which affected performance significantly as multiple documents are included in the verification process. A root certificate has no provenance whereas an intermediate does, pointing to some previous signer and its document.

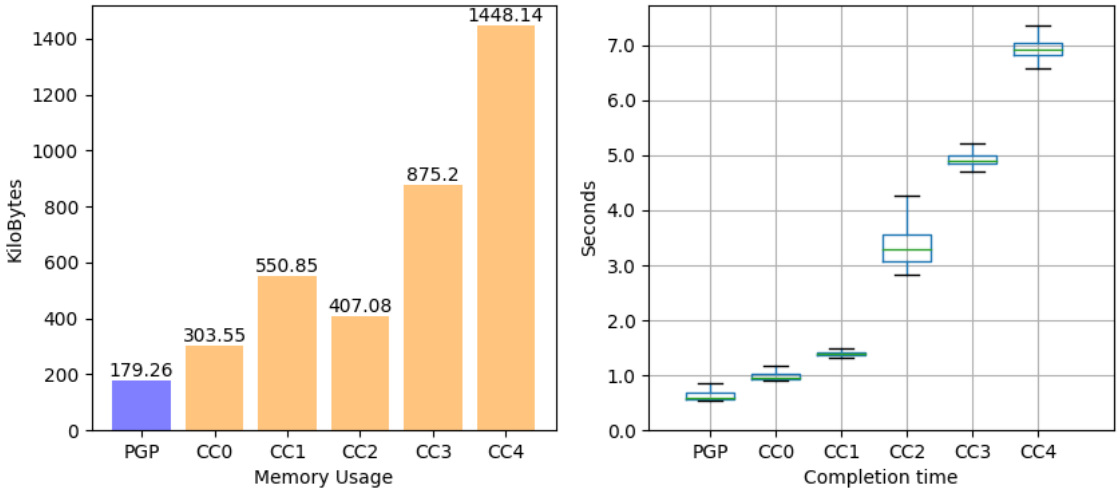


Figure 6.5: Notice how the PGP verification stays constant whereas CC verification grows linearly with chain length. It has a worst case complexity of $O(n)$ with n the actual chain length.

Conclusion

Verification in PGP is constant as only one document is involved in all cases. Clavichain's verification process depends on the upper and actual chain length as multiple documents are potentially used. Resource requirements increase linearly for longer chains even when not accounting for the network latency of fetching the documents. Later experiments suggest linked chains lose their effectiveness in propagating keys as more documents are involved in verification.

6.3.3 Management

Management deals with operations for the collection of certificates: be it in determining the missing ones or adding them to some body. Each model has their own implementation to add certificates and compute missing ones.

Adding Certificates

Adding certificates in ClaviChain means computing a new certificate for other nodes to extend. It uses the sign-operation of section 6.3.1 under the hood. PGP extracts the key and adds the certificate to the displayed collection.

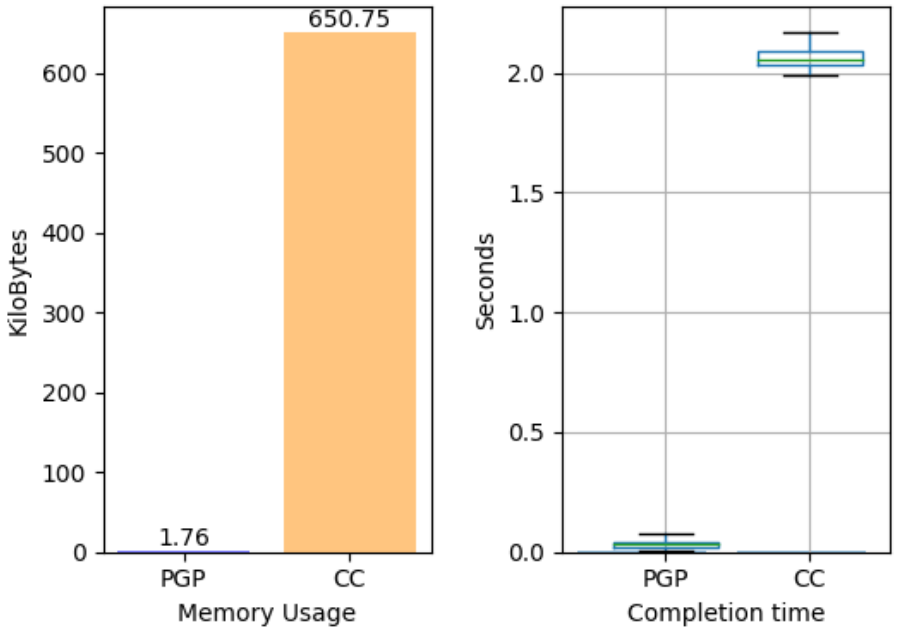


Figure 6.6: Adding certificates in ClaviChain is a more complex operation compared to PGP because a certificate needs to be signed to let other nodes extend the chain further. In ClaviChain, every participant which extends a chain is expected to store the resulting new certificate. The sign-operation of section 6.3.1 is used.

Computing Missing Certificates

This is a task complementary to adding certificates. It is performed every time a node checks its trusted neighbours for new certificates. ClaviChain has the additional worry of identifying the certificates that are deemed replaceable because of some detected inconsistency in provenance.

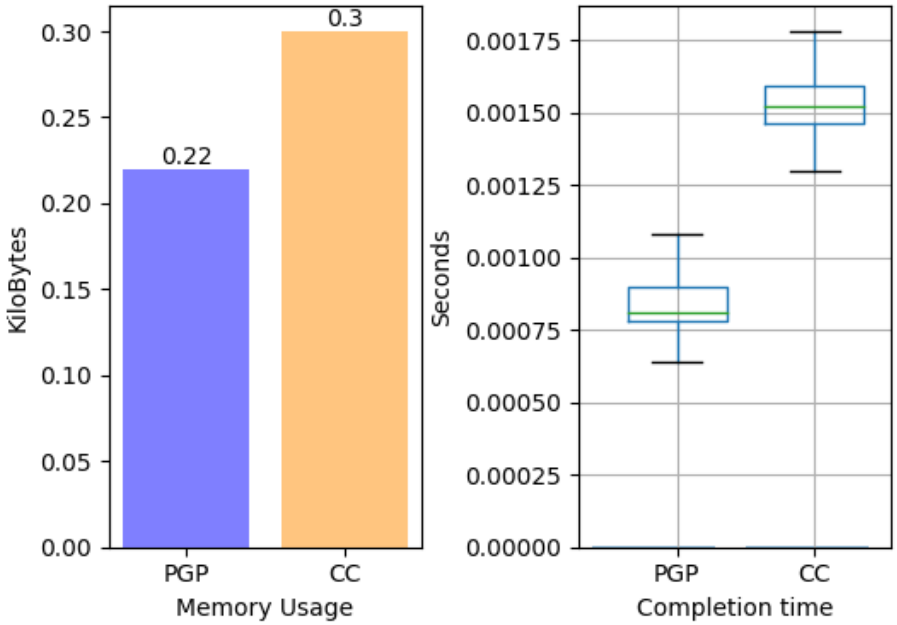


Figure 6.7: ClaviChain computes the missing certificates but also checks for documents which are deemed replaceable. The additional mapping makes for an insignificant difference in completion time measured in milliseconds. Completion time for PGP is on average 0.0008 seconds whereas for CC it is around 0.0015 seconds. Very low execution times in both cases.

Conclusion

Adding a certificate in ClaviChain has higher consumption because under the hood it calls a sign operation to make a new certificate. *Add* has a fixed amount of work. Computing missing certificates in the two implementations is insignificant. An explanation for this, is the lack of started nodes on the side of which the missing certificates are computed.

6.3.4 Summary

Including provenance in the certificate format makes some model-specific methods display considerably higher resource requirements. Signing requires twice the memory usage and thrice the completion time with verification growing linearly with chain length. Adding certificates has a higher resource usage in ClaviChain because it signs a new certificate every time the action is performed. ClaviChain has worse resource consumption for the same key propagation in comparison with PGP. An advantage is reached by increasing chain length thus improving propagation.

6.4 Certificate Propagation

Methods benchmarked in section 6.3 implement custom behaviour for the ClaviChain and Pretty Good Privacy implementations. This section examines their runtime properties in distributed systems, measurements focus on the propagation of certificates with and without provenance. Benchmarking the verification process of ClaviChain required the use of linear trust graphs to fetch the certificate chain iteratively. Experiments in this section make use of cycle graphs and strongly regular graphs to represent trust relations.

6.4.1 Effect of Network Size

The first experiment measures certificate propagation in networks with a variable numbers of participants. Measurements were performed on the *Firefly* server. Between 25 and 150 virtual nodes were started as to study the runtime properties of ClaviChain and Pretty Good Privacy. The spread and coverage metrics were used as to show some more qualitative information. As to mimic PGP more closely, the upper chain length of ClaviChain is set at 2. A chain then consists of 3 parties: an owner, intermediate *signer* and recipient of its certificate.

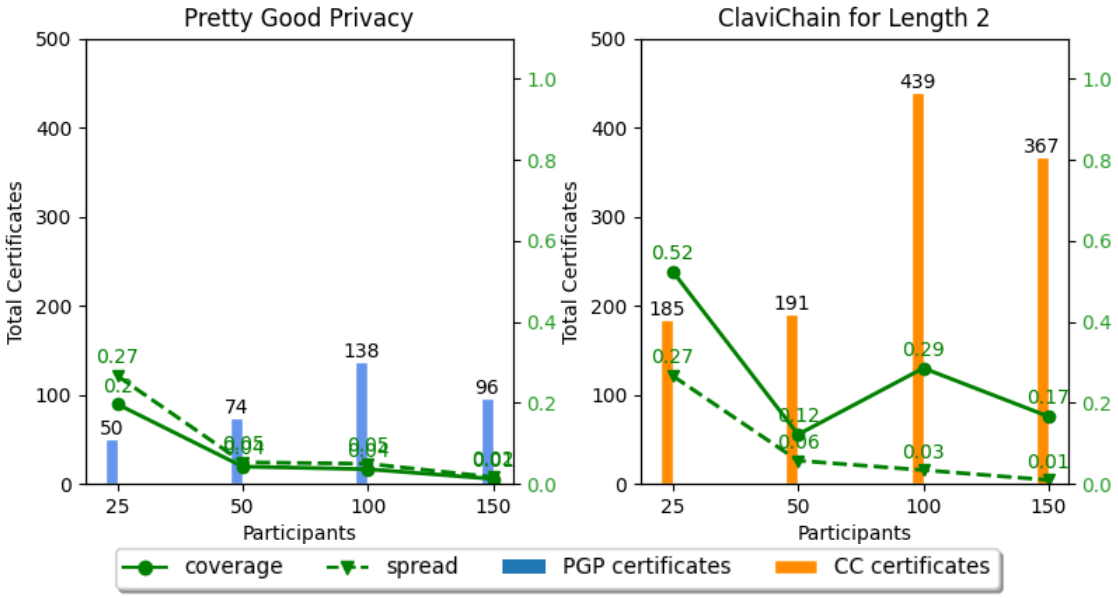


Figure 6.8: The number of certificates on the network differs between the models for the same spread metric. Every participant in possession of a key needs to store a certificate binding that key to a proof-of-ownership. Having one certificate belonging to the chain means a participant can verify all its members which makes for better coverage.

Coverage gives a notion of certificate predictability whereas spread stands for the measure in which participants obtain the other players' keys. Both the spread and coverage metric are correlated as high coverage implies high spread. Which makes sense as networks where each participants has all its peers' keys is able to do predictions on any certificate binding a key.

Conclusion

ClaviChain has higher coverage compared to PGP for the same spread when chain length is two. A chain length of two means only the following nodes are included when sharing keys: the owner, one signer and the recipient of its certificate. If one certificate can be verified, all members of its provenance chain are verifiable. A node can do predictions on much more certificates with provenance compared to PGP which is limited to current document. This makes ClaviChain much more flexible as more certificates can be consulted when searching a key.

6.4.2 Effect of Chain Length

Chain length affects the propagation of certificates by limiting the number of allowed intermediates. For this experiment, we measure its effect in a network of fixed size set at 50 participants. Measurements were sampled when coverage reached a stable value for 30 seconds in the set-up and were repeated a total of 30 times.

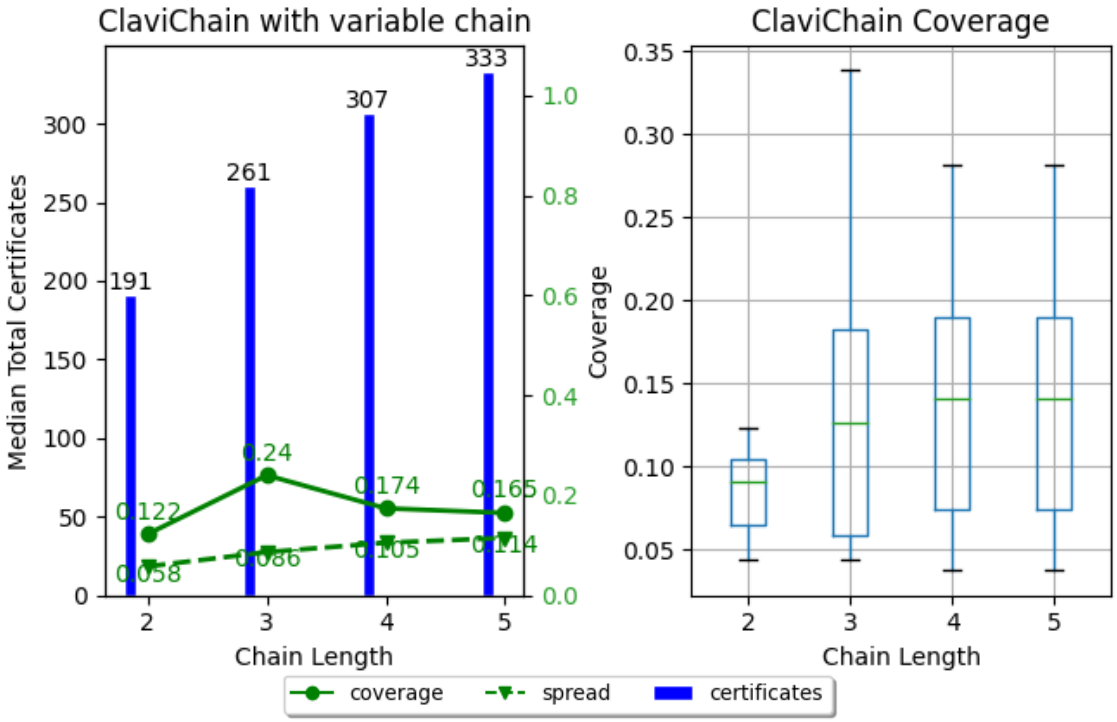


Figure 6.9: Figure left takes shows the median spread metric for different sizes. Key propagation is improved by elongating the chain. Spread is increased two-fold when doubling the chain length. Figure right shows the distribution of coverage stabilizes on the long run. Longer chains do not contribute significantly to higher coverage measurements due to the time requirements for propagation.

Spread and coverage improve in ClaviChain for chains longer than two. Verification times increase due to the longer provenance as more certificates are included in the verification process. Propagation therefore stagnates for longer chains because of the added workload.

Conclusion

Benchmarking revealed how adding provenance to the certificate format makes signing and verifying certificates significantly more expensive. Extending the chain length in ClaviChain improves coverage and spread although the effect is less significant for longer chains. This is because verification takes more time as more certificates need to be verified until the root. This slows propagation down. Note that this set-up assumes an ideal environment with no malicious behaviour which will be introduced and explored in next section.

6.5 Robustness

A trust model as a security measure should be robust enough to handle malicious participants. Experiments are set to measure the effect of malicious participants on propagation of forged documents. Provenance is an additional mechanism for its detection. The independent variables in experiments are the amount of malicious nodes and the number of shared neighbours of non-adjacent nodes (μ). Participants are set at 50 with experiments repeated 30 times.

6.5.1 Detection

Malicious participants in this set-up falsify certificates when signing. Root certificates with no previous signers are never forged in our set-up. Given a cycle graph discussed in 6.2.2, we change the percentage of malicious nodes to see its effect on certificate propagation.

Forgers falsify their displayed certificates and their presence is set at 0, 25, 50 and 75 percent of the total participants. ClaviChain's chain length is set at 2 to mimic PGP more closely. Figures below show the rate of falsified certificates in groups of 50 nodes. The percentage of forgers is a variable parameter.

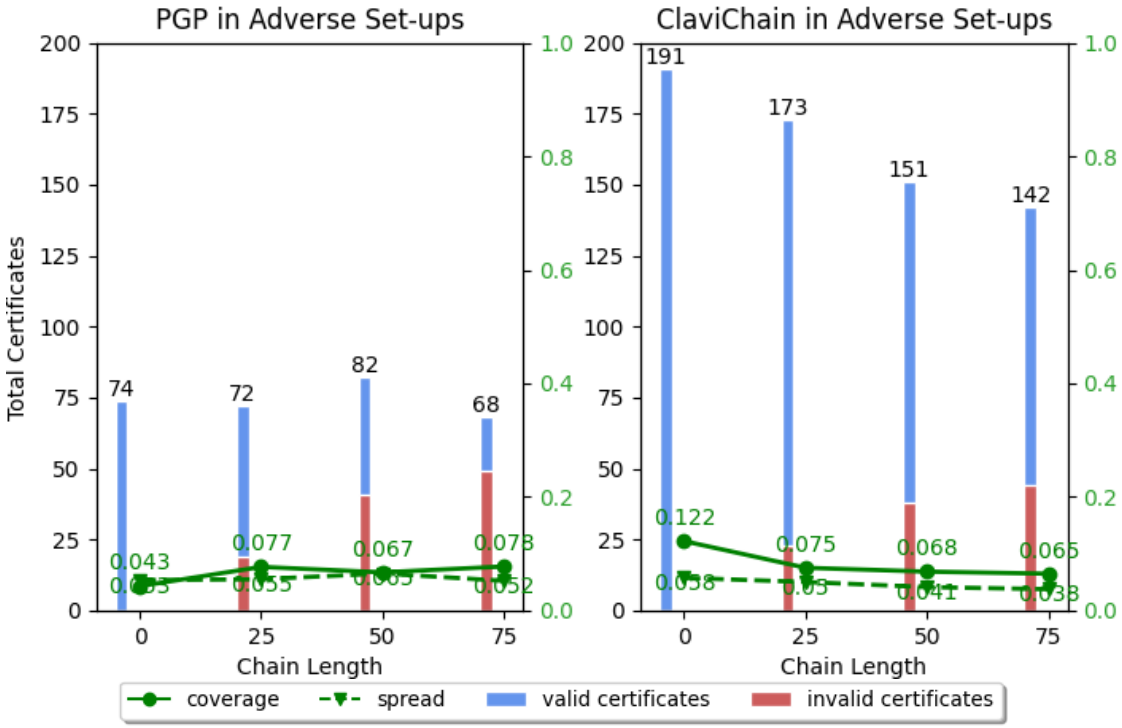


Figure 6.10: PGP is unable to detect forgery committed by signers deemed trustworthy and the amount of forged certificates rises with the rate of forgers. ClaviChain is able to detect forgery by signers and has greater accuracy with declined coverage.

Conclusion

Coverage in ClaviChain is lowered when increasing the rate of forgers in a cycle graph. The model is able to correctly detect forgery and has greater accuracy as a result which is indicated by the smaller slice of forged documents. PGP is not able to detect forgery, coverage and spread metrics remain unaffected as the percentage of falsified certificates in circulation increases. Spread in ClaviChain is negatively influenced by malicious forgers as no alternative issuers are available in the cyclic graph. Spread is improved by introducing more alternative signers made evident in section 6.5.2.

6.5.2 Effect of Alternative Signers

Provenance allows a node to detect forgery committed by intermediate signers which is not possible in PGP as it assumes trusted nodes to be reliable. When alternative issuers are lacking, propagation in ClaviChain is hindered as forgers will not have their certificates used.

Figures below show what effect multiple issuers have on coverage even in the presence of forgers in both CC and PGP. Provenance makes it possible to distinguish reliable issuers from forgers improving accuracy. Independent variables are the μ parameter for shared neighbours of non-adjacent nodes and the rate of malicious participants.

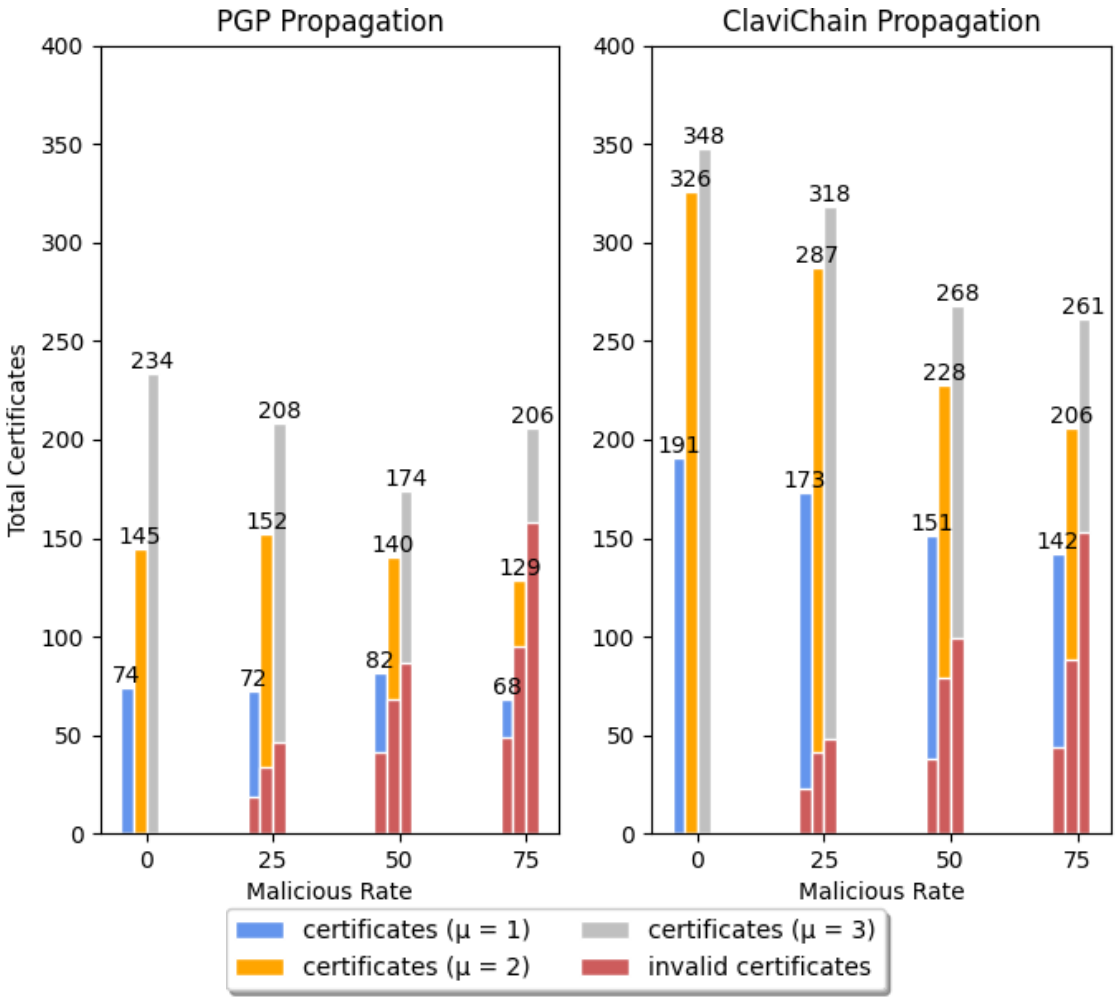


Figure 6.11: Amount of forged documents increases with the rate of forgers in both models. Forgers are not detected in PGP which means the rate of forged certificates mirrors the rate of malicious participants: 25 % forged documents for 25 % forgers, etc.
Certificates with provenance can detect forgery by intermediate issuers. In a set-up with few alternative issuers, the rate of forged documents remains low and the totality of certificates decreases with the rate of forgers.

Introducing more potential issuers (increased μ) increases forged documents in circulation for both models. The accuracy metric remains high in ClaviChain which means the model effectively detects forgery by intermediates, the same thing cannot be said about PGP.

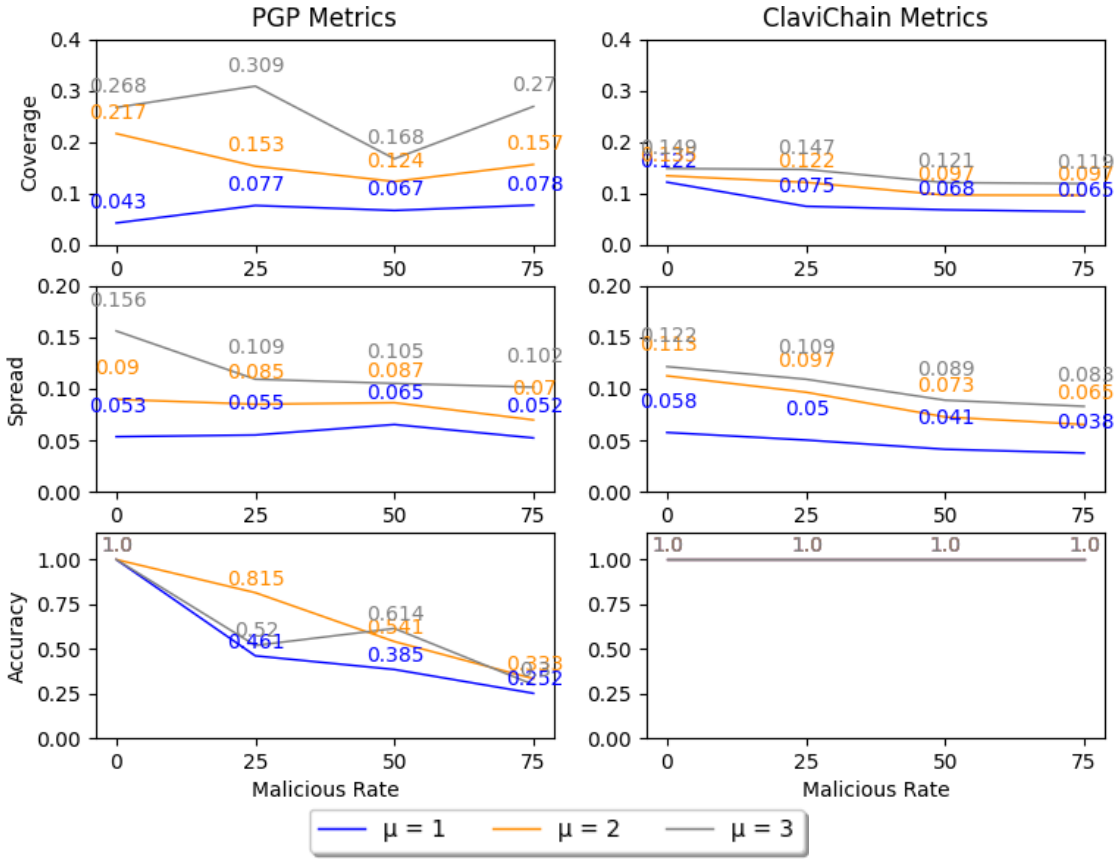


Figure 6.12: Forgery is detected in ClaviChain as indicated by high accuracy metrics. PGP has more unstable results for varying μ parameters as it cannot detect malicious issuers. Coverage is a metric that is unstable for PGP because forgers are not detected, falsified certificates flow freely. ClaviChain has a coverage and spread which decrease with the rate of forgers as they are detected in the model. Accuracy remains high in ClaviChain as provenance is used to detect malicious participants.

Conclusion

By including provenance in the certificates, ClaviChain is able to detect forgery committed by intermediate signers. PGP's effectiveness hinges on the assumption that the trusted participants are indeed reliable signers. Subverting this assumption undermines the integrity of the whole trust model. Coverage in ClaviChain is lowered as a consequence when the number of unreliable signers increases. Introducing alternative signers improves the spread in the network even in presence of forgers. Accuracy is very high in ClaviChain which makes for a robust model.

6.6 Threats to Validity

Coverage is one of the metrics used in our evaluation and determines the extent to which nodes are able to predict certificate correctness [18]. Coverage differs between the models due to the differing amounts of certificates for the same key propagation. In a transaction of certificates involving two parties and an intermediate, only one certificate is created by the signer in PGP. ClaviChain makes 3 of them in comparison: a root certificate at the owner's side, a first intermediate certificate and a second intermediate certificate upon accepting. We therefore included spread as an additional metric which determines the extent in which a node has the keys of its network participants. Taken together, the different coverage rates are seen as relatively equal in PGP and CC when the spread metric between the two is equal.
The amount of participants in the network and the number of repetitions of the experiments are points of contention of our experiments. 30 repetitions might not be adequate for experiments performed in distributed systems following the Actor model. Experiments involving the detection of forgery and effect of alternatives gave some inconsistent results for PGP as it is unable to detect forgery. Where in ClaviChain the results stabilized quickly due to the isolation of forgers, certificates in PGP were passed freely on the network meaning the metrics took a long time to stabilize and were not always consistent.

6.7 Conclusion

In this dissertation, a new certificate format was proposed with provenance fields allowing for signer-of-signer relations to emerge. The model using the format is called Web of Trust and its simulation program ClaviChain. Effects of provenance on coverage, spread and accuracy were measured in different scenarios.
From experiments, we made the following findings on ClaviChain:

- **Increased Resource Consumption** operations on certificates with provenance have a significantly higher resource consumption. Provenance doubles the memory usage and triples completion time when signing. Verification is more resource-demanding with root certificates having twice the memory usage and completion time of PGP certificates. Verifying intermediate certificates is linear in its complexity as it is correlated with the length of provenance. All in all, the format has higher resource requirements.
- **Increased Coverage for Similar Spread** dynamic program analysis first measured models in an ideal environment with no forgers. Coverage and spread decrease in the models with increased network size as they tend towards the same values. For an identical spread metric, the coverage of ClaviChain is much higher which means that overall more certificates are verifiable. A node has more choices when searching for certificates which makes propagation more flexible.
- **Chains are Less Effective with Increased Length** increasing chain length beyond two improves spread in the network by allowing for more than one signer between parties exchanging keys. Propagation is improved but its effectiveness tends to stagnate for longer chains as more intermediate certificates increase total verification time. Propagation tends to be slower for longer provenance chains.
- **More Robust Model** forgery by intermediate signers –not by root owners– is detected in ClaviChain. For set-ups with limited available signers, the spread of keys takes a hit but estimates on certificates remain very accurate. A way to improve propagation in such cases is to introduce more potential signers as the forgers are detected and isolated. Due to the better detection of forgeries, ClaviChain is said to be much more robust than PGP.

ClaviChain demands more resources for its certificate operations compared to PGP. A trade-off is increased coverage for the same spread and the ability to increase both metrics with prolonged chains even as they become less significant for longer chains. Forgery by intermediates is detected with provenance. Related work claimed more robust trust models are also more complex in nature [6]. ClaviChain is a more robust alternative on PGP with higher resource consumption.

7 Conclusion

The Web of Trust uses intermediate signers to open secure channels and its most known application is the PGP software for email encryption. It is limited by a lack of trust propagation mechanisms as participants trust direct signers but not those thereafter. Trust is not transitive and indirect trust relations are not possible. Only one signer is allowed between parties and relaxing this constraint increases the risk of forgery in the sequence of cooperating nodes.

Web of Trust has been successfully applied to email encryption but above-mentioned problems hamper its effectiveness in Internet-of-Things settings. In the use case of a decentralized Uber-like taxi service, the lack of signer-of-signer relations hampers key propagation. Web of Trust is therefore difficult to deploy in dynamic IoT networks without the use of multiple signers. To improve its usability in dynamic networks, this dissertation proposed ClaviChain as a novel certificate format with provenance. Its trust model called Web of Chains links certificates together into a provenance chain. Trust is propagated to indirect nodes by verifying the chain's integrity, allowing for multiple signers between two parties. Forgery is detected when provenance between certificates is inconsistent.

Empirical evaluation measured the performance of model-specific methods and of distributed systems abiding to Web of Trust or Web of Chains. Systems were examined in ideal scenarios and more realistic ones with forgers present. The Web of Chains model improves key propagation by allowing for multiple intermediate signers. Certificates are linked together with provenance and if one document is verifiable all previous ones are also verifiable which improves flexibility. Web of Chains is more robust against forgers compared to the Web of Trust model. Forgery by intermediate signers is detected in Web of Chains by verifying the added provenance. The additional provenance fields make for a considerably higher overhead for certificate operations. Verification time increases with longer provenance chains as more intermediates are included: key propagation is less effective for longer chains.

This dissertation brought forward a new certificate format with provenance and a simulation in Elixir of existing Web of Trust model and Web of Chains. Evaluation gave some interesting insights about the inclusion of provenance in the Web of Trust model. Although resource requirements are higher, Web of Chains allows for a few more additional signers between parties. It makes for a more robust trust model better suited for use in decentralized networks.

7.1 Future Work

This dissertation proposed a novel certificate format with provenance, allowing for multiple signers between a key owner and receiver. For all of our obtained results, some additional shortcomings were identified but not researched further due to time constraints:

- **Hybrid Certificate -and Behavior-based Approach** nodes in experiments only collect one certificate for verification. When multiple certificates are to be collected, a node needs to decide which provenance chain to extend. Behavior-based trust is a way to select the best chain for extension based on previous experience and reputation.
- **Chain Replacement Policies** if a provenance chain has inconsistencies in one of its records, the sequence needs to be recomputed from the mistake on wards. Forgery in experiments happens instantly with no time for new records to be added in the chain. Different replacement policies to repair a chain have been imagined ranging from selecting the shortest valid chain to the one with the best combined reputation of its members.

7.2 Final Remarks

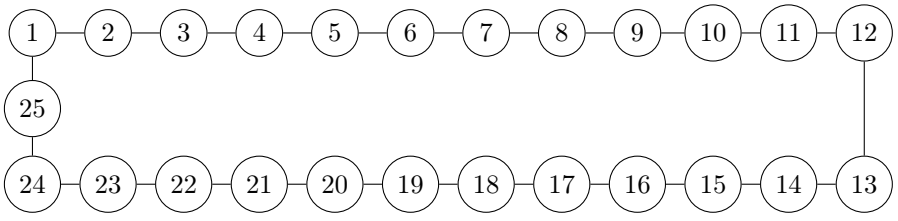
The new certificate format doubles as a provenance record linked with pointers to other certificates. Hashes are added for integrity verification in between records. When applied in Internet-of-Things settings, a few additional signers between participants is useful to open more secure channels. Web of Chains is a solution to secure distributed networks provided the resources are available for the additional overhead. Web of Trust is made more robust with provenance at the expense of its resource requirements.

A

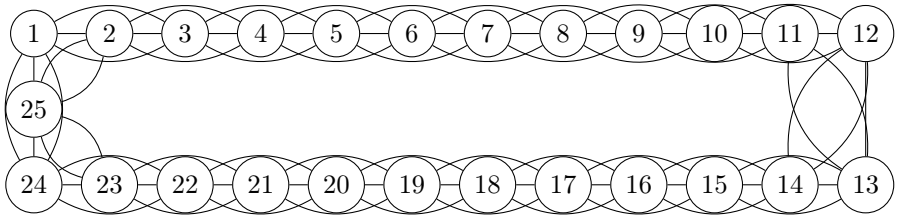
Appendix A: Trust Distributions

In this section, an overview is provided of the trust distributions employed in evaluation. They are illustrated as graphs. Benchmarking did not perform any runtime program analysis on groups of nodes so all figures below are part of experiments using profiling and monitoring techniques. The scope of experiments range between 25 and 150 participants. Only distributions with up to 25 participants are illustrated but they are trivially expanded to any size.

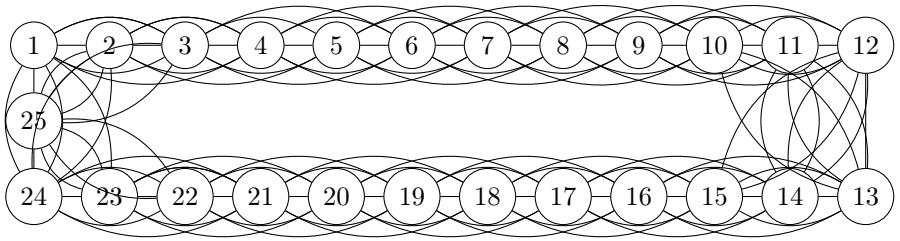
Listing A.1: Scenario 1: Simple Linear



Listing A.2: Scenario 2: Cycle Graph



Listing A.3: Scenario 3: Strongly Regular Graph ($\mu = 2$)



Listing A.4: Scenario 3: Strongly Regular Graph ($\mu = 3$)

B

Appendix B: Experimental Results

This appendix summarizes results of evaluation in tables. Benchmarking results are given in terms of memory usage and completion time, with the latter described as a 5-number summary. Profiling and monitoring results are given in terms of quantitative information (network size, propagated certificates, etc.) and qualitative information: coverage, spread and accuracy.

PGP	Memory Usage	Completion Time			
		Max	Min	Med	Avg
Sign	273.05	4.5550	1.9683	2.1005	2.1924
Verify	179.26	1.5295	0.5456	0.6016	0.6521
Add	1.76	0.1704	0.0065	0.0295	0.0316
Miss	0.22	45.1201	0.0006	0.0008	0.0009

Table B.1: PGP Benchmark Results.

ClaviChain	Memory Usage	Completion Time			
		Max	Min	Med	Avg
Sign (root)	407.08	5.2676	2.3537	2.4417	2.5889
Sign (inter)	666.76	7.6995	3.0984	3.3203	3.3613
Verify (0)	303.55	2.1124	0.8965	0.9510	1.0356
Verify (1)	550.85	5.6644	0.9105	0.9586	0.9999
Verify (2)	407.08	16.9769	2.8199	3.3016	3.6189
Verify (3)	875.2	15.8951	4.7116	4.9081	5.1973
Verify (4)	1448.14	18.6556	6.5904	6.9119	3.6189
Add	650.75	4.1852	1.9905	2.0568	2.0684
Miss	0.3	45.9727	0.0013	0.0015	0.0017

Table B.2: ClaviChain Benchmark Results.

	PGP				ClaviChain			
Participants	25	50	100	150	25	50	100	150
Certificates	50	74	138	96	185	191	439	367
Accuracy	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Coverage	0.1968	0.0429	0.0364	0.0125	0.5243	0.1222	0.2861	0.1665
Spread	0.2667	0.053	0.050	0.0169	0.2667	0.0576	0.0342	0.0097

Table B.3: PGP and ClaviChain for variable network size and chain length 2.

Chain Length	Certificates	Accuracy	Spread	Coverage			
				Max	Min	Med	Avg
2	191	1.0	0.0576	0.4506	0.0436	0.0906	0.1045
3	261	1.0	0.0861	0.3386	0.0434	0.1260	0.1277
4	307	1.0	0.1049	0.2813	0.0374	0.1405	1380
5	333	1.0	0.1145	0.2954	0.0374	0.1438	0.1479

Table B.4: ClaviChain for 50 participants with variable chain lengths.
Note: median value for spread metric is used.

	PGP			ClaviChain Length 2		
μ	1	2	3	1	2	3
$Accuracy_0$	1.0	1.0	1.0	1.0	1.0	1.0
$Accuracy_{25}$	0.4613	0.8147	0.5196	1.0	1.0	1.0
$Accuracy_{50}$	0.385	0.5413	0.6144	1.0	1.0	1.0
$Accuracy_{75}$	0.2517	0.3334	0.3	1.0	1.0	1.0
$Coverage_0$	0.0429	0.2168	0.2676	0.1222	0.1347	0.1489
$Coverage_{25}$	0.0768	0.1533	0.3092	0.0753	0.1470	0.1219
$Coverage_{50}$	0.0672	0.1236	0.1678	0.0685	0.1213	0.0970
$Coverage_{75}$	0.0776	0.1566	0.2696	0.0648	0.1195	0.0967
$Spread_0$	0.0535	0.0898	0.1563	0.0580	0.1127	0.1216
$Spread_{25}$	0.0551	0.0850	0.1094	0.0502	0.0967	0.1094
$Spread_{50}$	0.0653	0.0865	0.1053	0.4122	0.0727	0.0890
$Spread_{75}$	0.0522	0.0698	0.1016	0.0376	0.0653	0.0830

Table B.5: PGP and ClaviChain monitoring results.

Bibliography

[1] 11 profiling. `http://erlang.org/doc/efficiency_guide/profiling.html`. Accessed: 2021-02-27.

[2] Alfarez Abdul-Rahman. The pgp trust model. In *EDI-Forum: the Journal of Electronic Commerce*, volume 10, pages 27–31, 1997.

[3] David Adrian, Karthikeyan Bhargavan, Zakir Durumeric, Pierrick Gaudry, Matthew Green, J Alex Halderman, Nadia Heninger, Drew Springall, Emmanuel Thomé, Luke Valenta, et al. Imperfect forward secrecy: How diffie-hellman fails in practice. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 5–17, 2015.

[4] Monika Agrawal and Pradeep Mishra. A comparative survey on symmetric key encryption techniques. *International Journal on Computer Science and Engineering*, 4(5):877, 2012.

[5] Ejaz Ahmed, Ibrar Yaqoob, Abdullah Gani, Muhammad Imran, and Mohsen Guizani. Internet-of-things-based smart environments: state of the art, taxonomy, and open research challenges. *IEEE Wireless Communications*, 23(5):10–16, 2016.

[6] Efthimia Aivaloglou, Stefanos Gritzalis, and Charalabos Skianis. Trust establishment in ad hoc and sensor networks. In *International Workshop on Critical Information Infrastructures Security*, pages 179–194. Springer, 2006.

[7] Efthimia Aivaloglou, Stefanos Gritzalis, and Charalabos Skianis. Trust establishment in sensor networks: behaviour-based, certificate-based and a combinational approach. *International Journal of System of Systems Engineering*, 1(1-2):128–148, 2008.

[8] Efthimia Aivaloglou, Stefanos Gritzalis, and Charalabos Skianis. Trust establishment in sensor networks: behaviour-based, certificate-based and a combinational approach. *International Journal of System of Systems Engineering*, 1(1-2):128–148, 2008.

[9] Abdullah Al Hasib and Abul Ahsan Md Mahmudul Haque. A comparative study of the performance and security issues of aes and rsa cryptography. In *2008 Third International Conference on Convergence and Hybrid Information Technology*, volume 2, pages 505–510. IEEE, 2008.

[10] Adel Alkhalil and Rabie A Ramadan. Iot data provenance implementation challenges. *Procedia Computer Science*, 109:1134–1139, 2017.

[11] Mohamed Ben-Daya, Elkafi Hassini, and Zied Bahroun. Internet of things and supply chain management: a literature review. *International Journal of Production Research*, 57(15-16):4719–4742, 2019.

[12] Subhrabrata Choudhury, Suman Deb Roy, and Sneha Aman Singh. Trust management in ad hoc network for secure dsr routing. In *Novel algorithms and techniques in telecommunications, automation and industrial electronics*, pages 496–500. Springer, 2008.

[13] Carlton R Davis. A localized trust management scheme for ad hoc networks. In *Proceedings of the 3rd International Conference on Networking (ICN'04)*, pages 671–675, 2004.

[14] Christophe De Troyer, Jens Nicolay, and Wolfgang De Meuter. Building iot systems using distributed first-class reactive programming. In *2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 185–192. IEEE, 2018.

[15] Loic Etienne. Malicious traffic detection in local networks with snort. Technical report, 2009.

[16] Mohamed Firdhous. Implementation of security in distributed systems-a comparative study. *arXiv preprint arXiv:1211.2032*, 2012.

[17] Arthur Gervais, Ghassan O Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. On the security and performance of proof of work blockchains. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 3–16, 2016.

[18] Guibing Guo, Jie Zhang, and Julita Vassileva. Improving pgp web of trust through the expansion of trusted neighborhood. In *2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, volume 1, pages 489–494. IEEE, 2011.

[19] Sergio Gusmeroli, Salvatore Piccione, and Domenico Rotondi. Iot access control issues: a capability based approach. In *2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 787–792. IEEE, 2012.

[20] Mikael Gustavsson and Jan Ljungberg. Uber and the swedish taxi market: A discourse analysis. 2020.

[21] Darrel Hankerson, Alfred J Menezes, and Scott Vanstone. *Guide to elliptic curve cryptography*. Springer Science & Business Media, 2006.

[22] Eman Salim Ibrahim Harba. Secure data encryption through a combination of aes, rsa and hmac. *Engineering, Technology & Applied Science Research*, 7(4):1781–1785, 2017.

[23] Thomas Hardjono and Lakshminath R Dondeti. *Security in Wireless LANS and MANS (Artech House Computer Security)*. Artech House, Inc., 2005.

[24] Ragib Hasan. Protecting the past and present of data, with applications in provenance and regulatory-compliant databases. In *Proceedings of the Third SIGMOD PhD Workshop on Innovative Database Research (IDAR 2009)*. Citeseer, 2009.

[25] Ray Hunt. Pki and digital certification infrastructure. In *Proceedings. Ninth IEEE International Conference on Networks, ICON 2001.*, pages 234–239. IEEE, 2001.

[26] Uzair Javaid, Muhammad Naveed Aman, and Biplab Sikdar. Blockpro: Blockchain based data provenance and integrity for secure iot environments. In *Proceedings of the 1st Workshop on Blockchain-enabled Networked Sensor Systems*, pages 13–18. ACM, 2018.

[27] Yixin Jiang, Chuang Lin, Xuemin Shen, and Minghui Shi. Mutual authentication and key exchange protocols for roaming services in wireless mobile networks. *IEEE Transactions on Wireless Communications*, 5(9):2569–2577, 2006.

[28] Qi Jing, Athanasios V Vasilakos, Jiafu Wan, Jingwei Lu, and Dechao Qiu. Security of the internet of things: perspectives and challenges. *Wireless Networks*, 20(8):2481–2501, 2014.

[29] Prasad Jogalekar and Murray Woodside. Evaluating the scalability of distributed systems. *IEEE Transactions on parallel and distributed systems*, 11(6):589–603, 2000.

[30] Raj Kamal Kapur and Sunil Kumar Khatri. Secure data transfer in manet using symmetric and asymmetric cryptography. In *2015 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO)(Trends and Future Directions)*, pages 1–5. IEEE, 2015.

[31] Rohit Khare and Adam Rifkin. Weaving a web of trust. *World Wide Web Journal*, 2(3):77–112, 1997.

[32] Neal Koblitz, Alfred Menezes, and Scott Vanstone. The state of elliptic curve cryptography. *Designs, codes and cryptography*, 19(2-3):173–193, 2000.

[33] Paul C Kocher. On certificate revocation and validation. In *International conference on financial cryptography*, pages 172–177. Springer, 1998.

[34] Jiejun Kong, Z Petros, Haiyun Luo, Songwu Lu, and Lixia Zhang. Providing robust and ubiquitous security support for mobile ad-hoc networks. pages 251–260, 2001.

[35] Shancang Li, Li Da Xu, and Shanshan Zhao. The internet of things: a survey. *Information Systems Frontiers*, 17(2):243–259, 2015.

[36] Lourdes Lopez and Justo Carracedo. Hierarchical organization of certification authorities for secure environments. In *Proceedings of SNDSS'97: Internet Society 1997 Symposium on Network and Distributed System Security*, pages 112–121. IEEE, 1997.

[37] Tyler M Masthay. An introduction to the elixir programming language. 2017.

[38] Chetan N Mathur and KP Subbalakshmi. Digital signatures for centralized dsa networks. In *2007 4th IEEE Consumer Communications and Networking Conference*, pages 1037–1041. IEEE, 2007.

[39] Lik Mui, Mojdeh Mohtashemi, and Ari Halberstadt. A computational model of trust and reputation. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, pages 2431–2439. IEEE, 2002.

[40] Irene CL Ng and Susan YL Wakenshaw. The internet-of-things: Review and research directions. *International Journal of Research in Marketing*, 34(1):3–21, 2017.

[41] Radia Perlman. An overview of pki trust models. *IEEE network*, 13(6):38–43, 1999.

[42] Why I Wrote PGP. Philip zimmermann.

[43] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[44] N Ruangchaijatupon and P Krishnamurthy. Encryption and power consumption in wireless lans-n,". In *The Third IEEE workshop on wireless LANS*, pages 148–152, 2001.

[45] Swapna B Sasi, Dila Dixon, Jesmy Wilson, and P No. A general comparison of symmetric and asymmetric cryptosystems for wsns and an overview of location based encryption technique for improving security. *IOSR Journal of Engineering*, 4(3):1, 2014.

[46] Elaine Shi and Adrian Perrig. Designing secure sensor networks. *IEEE Wireless Communications*, 11(6):38–43, 2004.

[47] Sabah Suhail, Choong Seon Hong, M Ali Lodhi, Faheem Zafar, Abid Khan, and Faisal Bashir. Data trustworthiness in iot. In *2018 International Conference on Information Networking (ICOIN)*, pages 414–419. IEEE, 2018.

[48] Hui Suo, Jiafu Wan, Caifeng Zou, and Jianqi Liu. Security in the internet of things: a review. In *2012 international conference on computer science and electronics engineering*, volume 3, pages 648–651. IEEE, 2012.

[49] Alexander Ulrich, Ralph Holz, Peter Hauck, and Georg Carle. Investigating the openpgp web of trust. In *European Symposium on Research in Computer Security*, pages 489–507. Springer, 2011.

[50] Anila Umar, Muhammad Abbas, and Saad Rehman. Metrics and tools that are available fortesting scalability.

[51] Xinlei Wang, Kai Zeng, Kannan Govindan, and Prasant Mohapatra. Chaining for securing data provenance in distributed information networks. In *MILCOM 2012-2012 IEEE Military Communications Conference*, pages 1–6. IEEE, 2012.

[52] Douglas Brent West et al. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, 2001.

[53] Jongho Won, Ankush Singla, Elisa Bertino, and Greg Bollella. Decentralized public key infrastructure for internet-of-things. In *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*, pages 907–913. IEEE, 2018.

[54] Teng Xu, James B Wendt, and Miodrag Potkonjak. Security of iot systems: Design challenges and opportunities. In *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 417–423. IEEE, 2014.

[55] Andrea Zanella, Nicola Bui, Angelo Castellani, Lorenzo Vangelista, and Michele Zorzi. Internet of things for smart cities. *IEEE Internet of Things journal*, 1(1):22–32, 2014.

[56] Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Xiangping Chen, and Huaimin Wang. Blockchain challenges and opportunities: A survey. *International Journal of Web and Grid Services*, 14(4):352–375, 2018.