Institute of Information Security

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit

PKIs based on Blockchains

Damir Ravlija

Course of Study:	Informatik
Examiner:	Prof. Dr. Ralf Küsters
Supervisor:	Dipl.-Math. Mike Simon
Commenced:	November 6, 2018
Completed:	May 6, 2019

Abstract

Cryptographic protocols such as TLS rely on Public Key Infrastructure (PKI) to provide privacy to the users on the web. In traditional PKI a certain number of Certificate Authorities (CA) issue certificates which affirm that the CA verified the public key binding. However, since CAs in numerous cases behaved maliciously and issued unauthorized certificates, alternatives to traditional PKI model are being researched. Promising alternative is a blockchain technology which seems to be suitable for the implementation of PKIs: A blockchain is decentralized usually with only a few trust anchors. Data has to pass a consensus procedure before it becomes part of the state of the blockchain. Hence, blockchain offers a decentralized alternative to current CA-based PKI model.

In this thesis we survey the current state of research into PKIs based on blockchains. Firstly, we present PKI and blockchain, two integral parts of such systems. There we concentrate on PKI models and blockchain platforms that are relevant for the existing blockchain-based PKI proposals. We then introduce, classify, and present PKI systems based on blockchains. In the following chapter we discuss security properties, prospects for adoption, underlying blockchains, and distinctive features of blockchain-based PKI systems which are in the course of this compared to each other, to conventional PKIs, and its extensions. In the end, we introduce TKI, a PKI system developed on permissionless Ethereum blockchain that extends CA-based PKI and combines it with a Web of Trust architecture.

Kurzfassung

Kryptografische Protokolle wie TLS verlassen sich auf Public Key Infrastructure (PKI), um die Privatsphäre der Webbenutzer zu schützen. In der traditionellen PKI gibt es eine bestimmte Anzahl von Certificate Authorities (CA), die bestätigen, dass eine CA die Schlüsselbindung verifiziert hat. Da sich jedoch die CAs in zahlreichen Fällen bösartig verhalten haben, sowie unberechtigte Zertifikate ausgestellt haben, werden Alternativen zum traditionellen PKI Modell erforscht. Eine vielversprechende Alternative ist die Blockchain-Technologie, welche möglicherweise für die Implementierung der PKIs geeignet ist: Die Blockchain ist üblicherweise mit nur wenigen Vertrauensankern dezentralisiert. Daten müssen das Konsensverfahren durchgehen bevor sie Teil des Blockchain-Zustands werden. Aus diesem Grund bietet Blockchain eine dezentralisierte Alternative zum aktuellen CA-basierten PKI Modell.

In dieser Bachelorarbeit wird der jetzige Forschungsstand der blockchainbasierten PKIs untersucht. Zunächst präsentieren wir PKI und Blockchain, zwei wesentliche Bestandteile von solchen Systemen. Dabei konzentrieren wir uns auf die PKI Modelle und Blockchain-Platformen, die relevant für die existierenden blockchainbasierten PKI Systeme sind. Anschließend werden PKI Systeme auf Basis von Blockchains eingeführt, klassifiziert und präsentiert. Im folgenden Kapitel werden die Sicherheitseigenschaften, Adoptionsaussichten, darunterliegende Blockchains, und charakteristische Merkmale der blockchainbasierten PKI Systeme diskutiert, und sie werden mit den konventionellen PKIs und ihren Erweiterungen verglichen. Zuletzt stellen wir TKI vor, ein PKI-System, das auf der permissionless Ethereum Blockchain entwickelt wurde, und welches die CA-basierte PKI erweitert und mit Web of Trust Architektur kombiniert.

Contents

List of Figures

List of Tables

List of Listings

List of Abbreviations

ASIC Application-specific integrated circuit. 42

CA Certificate Authority. 17

CRL Certificate Revocation List. 24

CT Certificate Transparency. 31

DANE DNS-Based Authentification of Named Entities. 33

DoS Denial of Service. 27

DV Domain Validated. 25

EV Extended Validation. 25

IoT Internet of Things. 45

IPFS InterPlanetary File System. 52

MitM Man-in-the-middle. 22

OCSP Online Certificate Status Protocol. 24

OV Organization Validated. 25

PGP Pretty-Good-Privacy. 28

PKI Public Key Infrastructure. 17

PKP Public Key Pinning. 31

PoS Proof of Stake. 39

PoW Proof of Work. 38

STR Signed Tree Root. 30

TKI Transparent (Public) Key Infrastructure. 71

TLD Top-level Domain. 55

TLS Transport Layer Security. 17

WoT Web of Trust. 28

1 Introduction

1.1 Motivation

Security and privacy of users on the internet is nowadays assured through the usage of Transport Layer Security (TLS) protocol which uses asymmetric or public key cryptography to establish a secure connection between parties [33]. In public key cryptography every party has a public and private key which allow them to encrypt, decrypt, sign and validate messages. To be able to exchange messages securely with another user, one needs to know the public key of its communication partner. The user has to be sure that the public key which she uses for communication actually belongs to her communication partner and not some other potentially malicious entity. Goal of Public Key Infrastructure (PKI) is to provide a framework which assures the user that the link between a public key and identity of an entity that holds the corresponding private key is valid [93].

Traditional PKI employs trust in third parties. X.509, which is a PKI standard used most widely today, is based on trust in third parties called certificate authorities. Certificate authority (CA) verifies bindings between public keys and entities and accordingly issues digital certificates to entities. An entity can use a digital certificate to prove that a CA verified its public key binding [14].

Modern browsers and operating systems trust a certain number of CAs by default [46]. This makes current CA-based PKI model centralized because it centralizes trust to only those CAs that are trusted by browser and OS vendors. Further centralization happens through mergers and acquisitions of existing CAs. For example, DigiCert, one of the largest CAs acquired certificate authority businesses of Verizon [24], Symantec [34], and announced acquisition of QuoVadis [121].

Inherent assumption that the current PKI model makes is that CA entities are trustful and none of them will behave maliciously or issue unauthorized certificates. Since browsers trust CAs by default, every time a user visits a website which uses TLS, its privacy and security depend on the trustfulness of the CA which issued the certificate [46]. This presents a major weakness of the current PKI model because every CA represents a single-point of failure. Every CA included in the browser can issue any kind of certificate. There can be several certificates issued by multiple CAs for one key binding and all of these certificates would be considered trusted by the browser [38]. Each of the CAs included in the browser has to be absolutely trusted because a single rogue CA could wreak havoc. It could, for instance, issue an unauthorized certificate for *google.com* to some malicious entity which could in turn pretend to be Google. Connection to an entity with such certificate would be accepted by the user's browser.

Through the years there has been some criticism of traditional PKI system based on CAs. Holz et al. analyzed the web PKI environment in 2011 and concluded that "PKI [...] is in a sorry state" [58]. In 2000, Ellison and Schneier presented ten risks of PKI [42]. Risk #1 according to them was inadequately defined notion of "trust" in CAs. They also question usefulness of $(\textit{public key}, \textit{entity})$

bindings. Furthermore, they point out that CAs are actually not authoritative for information contained in the certificate such as domain names. Their skeptical attitude towards traditional CA-based PKI proved in the last few years to be justified.

In order to decrease the needed amount of trust in CAs, there exist multiple extensions for the current CA-based PKI model. For example, there have been several proposals for transparency logs such as CT [69], AKI [64], ARPKI [8]. Transparency log is a server on which CAs have to post every issued certificate allowing other entities to monitor CAs for misbehavior. Most widely deployed transparency log system is Google's Certificate Transparency (CT). Although Google has moved the deadline for enforcement of CT in Chrome multiple times, since 2018, Chrome accepts newly issued certificates only if they are posted to CT's transparency logs [25].

In the course of the last few years there were multiple major incidents that affected CAs across the world. Culprits behind the incidents were not always malicious actors that attacked the system, but there were also situations where CAs themselves made mistakes or misbehaved.

- In 2011 an attacker compromised a Dutch Certificate Authority DigiNotar [89]. On this occasion over 500 unauthorized certificates were issued. As a result of the attack DigiNotar went bankrupt.
- Symantec made several mistakes between 2009 and 2017. For example, between April 2009 and September 2015 Symantec issued test certificates which were not appropriately validated [21]. In an audit performed in 2015, it was revealed that Symantec misissued 164 certificates for 76 different domains [103]. Same audit also showed almost 2500 certificates issued for unregistered domains. As a consequence of this, with the release of Chrome 66 in 2016, certificates issued by Symantec before June 1, 2016 were distrusted by Chrome. Moreover, with the release of Chrome 70 all the remaining certificates issued by Symantec before its certificate business has been acquired by DigiCert are distrusted by Chrome [86].
- In 2016, Google started distrusting certificates issued by WoSign and StartCom. It was discovered that WoSign intentionally issued an unauthorized certificate for one of the Github's domains [117]. Furthermore, after buying StartCom, WoSign deliberately mislead community by presenting StartCom as an independent entity even though WoSign replaced StartCom's internal company structure and employees with their own.

Although there are many extensions for traditional CA-based PKI, all of the approaches still suffer from centralization. A blockchain technology appeared in 2008 as a fundamental building block underlying Bitcoin cryptocurrency [83]. Blockchain can be thought of as a distributed database on whose content participants in the network agree by running a consensus algorithm. It runs on a peer-to-peer network where participants distribute the current state of the blockchain through gossip protocol. Some blockchains also have an ability of running programs in a distributed fashion where all the users agree on the result of a program. Since it is based on consensus of participants that provides distributed append-only storage, blockchain appears to be suitable for implementation of PKI on top of it. Consensus on the contents of the blockchain's global state makes decentralized PKI possible. For the aforementioned reasons, blockchain-based PKIs are an active area of research with multiple proposed PKIs based on blockchains that either extend existing PKI models or introduce novel features made possible with a blockchain.

1.2 Thesis Structure

First two chapters deal mainly with the foundations necessary for the rest of the thesis. Following chapters deal with the main subject of the thesis, survey of existing blockchain-based PKIs and presentation of a system developed in the course of this thesis. Throughout the paper we aim to present a survey of PKIs based on blockchains. We present and compare the existing systems as well as look at their distinctive features. Moreover, we also present TKI, a blockchain-based PKI system that through the use of browser extension enables web users to see how the key bindings they encounter for a specific domain compare with bindings encountered by other users.

Chapter 2: PKI Firstly, we introduce public key cryptography and present a key binding problem that motivates the need for PKI. Secondly, we introduce PKI by presenting PKI models and standards currently employed in practice as well as models which blockchain-based systems use as a starting point. Some of the proposed extensions to CA-based PKI which do not use blockchain are looked into as well. In the course of this chapter we also look at vulnerabilities of some of the models.

Chapter 3: Blockchain After presenting PKI, we introduce blockchains. In the course of this we look at the blockchain's structure, its functionalities, types, and concepts that are associated with it. The emphasis in this chapter is placed on Bitcoin and Ethereum, the largest blockchain platforms on which most of the PKI systems are based. Likewise, attacks specific for the blockchain protocol are presented as well.

Chapter 4: PKI Systems based on Blockchains In this chapter we classify and present proposals for PKI systems that are based on blockchains.

Chapter 5: Discussion Here we discuss and compare blockchain-based PKI systems. We look at their security properties, discuss potential adoption of the systems and assumptions which they make, look at applications of the presented systems as well as the deployment costs. Additionally, we also discuss problems associated with the underlying blockchains and compare blockchain-based PKI systems with conventional PKI models and their extensions.

Chapter 6: TKI Transparent (public) key infrastructure system is presented in this chapter. TKI is a system that combines CA-based PKI and WoT architecture. It mitigates CA and user misbehavior through transparency and usage of blockchain as an unchangeable public storage. Likewise, it supports different types of certificates including client certificates.

Chapter 7: Conclusion Summary of the thesis and concluding remarks.

2 Public Key Infrastructure

This chapter introduces PKI, the main concept behind this thesis. Firstly, we look at the reason why public key cryptography is needed, and then present the key binding problem which is the main weakness of public key cryptography that motivated the development of PKI. This brings us to existing PKIs whose concepts and workings are looked into. Considering that CA-based PKI is predominantly used, inner workings of certificate handling by the CAs using the dominant certificate standard X.509 is explored somewhat more in detail. PGP and CONIKS as two PKI models which also present a foundation for many blockchain-based PKI systems are presented as well. Throughout the chapter and the rest of the thesis we will use fictional users Alice and Bob to help us present the encountered concepts.

Cryptographic primitives are building blocks for cryptographic protocols which allow Alice and Bob to securely communicate over a network that is perhaps controlled by an adversary. The specific cryptographic protocol deployed most widely on the internet to provide security is Transport Layer Security (TLS). It is used on the web in an HTTPS protocol where an application layer information contained in HTTP packets is secured with TLS.

Cryptographic primitives that are used in cryptographic protocols can be subdivided into two groups depending on keys that they use. There are symmetric cryptographic primitives that rely on the existence of a key k that is shared between two parties. Examples of such primitives are symmetric encryption scheme and MAC (Message Authentication Code), symmetric scheme similar to digital signatures. Another kind are asymmetric cryptographic primitives, examples of which are asymmetric encryption schemes and digital signatures (Appendix A.2 on page 93).

A scenario in which asymmetric cryptography is used might look as follows. Let us assume that Alice wants to send a message m to Bob. We assume also that at some earlier time Bob ran a generation algorithm which generated a pair of keys $(k, \hat{k})$ and sent k through a secure channel to Alice. Here k denotes a public key and $\hat{k}$ is its corresponding private key. At first, Alice encrypts a message m using Bob's public key k. To ensure integrity of the message Alice can use her own private key to sign the message. Afterwards, Alice sends a message with the signature to Bob. Following the receipt of the message, Bob checks its integrity by using Alice's public key to check the signature of the message. Thereafter, Bob decrypts the message using his private key.

These types of cryptographic primitives have differing advantages and disadvantages. For this reason, in practice cryptographic protocols such as TLS use a combination of both primitives. Symmetric cryptographic primitives are significantly more efficient, while, depending on an application, asymmetric primitives might be too inefficient. To exemplify, Gura et al. compared RSA and ECC asymmetric algorithms and found 1024-bit RSA encryption to be completely unsuitable for small processors [56]. Even so, Eisenbarth et al. found that ECC is 100 to 1000 times less efficient than AES, an algorithm used in symmetric cryptography [41].

However, when using symmetric primitives every link between two communication partners needs a different key. Hence, if there are n communication partners, n^2 different keys are needed for each of them to be able to communicate with all of the others. This makes symmetric primitives extremely impractical since every time one wants to communicate with a new communication partner, one needs a new key that has to be exchanged between communication partners.

Conversely, asymmetric primitives do not suffer from the key explosion problem because public key is not hidden which makes it available to all possible communication partners. In spite of that, there is another problem that affects asymmetric schemes. An attacker could pretend to be Bob and present her public key in a way that another network participant believes it is Bob's public key. For this reason, users need to be assured that they use the correct public key when communicating. That is, they need to verify (E, k), a binding between an entity E and public key k. This is called the key binding problem.

Nevertheless, having users check the ownership of every public key by themselves would be extremely impractical. Users would have to check a key binding of every network participant to whom they want to communicate. If a network participant changes its public key, it would again have to prove that the new public key belongs to it. Hence, without solution to a key binding problem public key cryptography would not bring significant improvements compared to symmetric primitives which suffer from a key explosion problem. To avoid this problem and make public cryptography usable in practice, Public Key Infrastructure (PKI) was developed.

2.1 Key Binding Problem

Goal of PKI is to offer a scalable solution to key binding problem. We therefore at first look at key binding problem. Key binding problem illustrates the main weakness of asymmetric cryptography. When Alice wants to send a message to Bob, she has to encrypt it using Bob's public key. However, how does she know that the public key which she wants to use actually belongs to Bob? In order to make asymmetric cryptography usable in practice, this problem has to be solved in a scalable way [66].

Key binding denotes a binding between an entity and its public key, and can be thought of as a pair (E, k), where E represents an entity which could be a domain or some other identifier and k represents a public key. The problem refers to the fact that individual users have to assure themselves that a public key which they use actually belongs to their communication partner and not an adversary. Since network protocols that do not use cryptographic protocols are insecure and prone to Man-in-the-middle (MitM) attacks, if Alice wants to prove to Bob that a certain public key belongs to her, she cannot simply send it to Bob because an adversary could intercept the message and change the public key contained within it. In this situation Bob would think that the received public key belongs to Alice even though it actually belongs to an adversary. Thus, if Bob tries to send or digitally sign a message intended for Alice using asymmetric cryptography, he would actually encrypt it with an adversary's public key which would allow an adversary to read the message. This would make digital signatures unreliable and pointless because any third party acting as an MitM could sign messages in somebody else's name, and non-repudiation property would not be ensured (Appendix A.2 on page 93).

Notion of digital certificates is often used in PKI. Trusted entities that verify key bindings can issue a digital certificate which is a document that can serve as a proof that the trusted entity indeed verified the key binding. In order to assure receivers of certificates that they indeed issued certificates, trusted entities digitally sign certificates using their private keys. There are numerous standards that define the format in which digital certificates can be represented. Examples of such standards are OpenPGP [48] and the most widely used certificate standard X.509 [14].

2.2 CA-based PKI

Nowadays most widely used PKI is based on trusted third parties called Certificate Authorities. HTTPS protocol which uses TLS to provide privacy to the users on the web relies on this type of PKI. More specifically, on X.509 standard that specifies how the certificates issued by CAs should look like [14].

An entity Alice that for instance runs a website and wants to offer its users the ability to connect securely to it using HTTPS at first generates a public and private key. Alice wants to assure her users that she indeed possesses the corresponding private key. Since her users trust CAs, she can do this by making a request to a CA to verify her key binding. CA verifies her key binding, creates a certificate, and signs the certificate using its private key. In return, Alice pays a fee to CA for this service. CA issues the created certificate to Alice with which CA claims that it verified Alice's key binding. Because users trust the CA unconditionally, Alice can use the issued certificate to prove to her communication partners that the public key indeed belongs to her.

CAs are, on the one hand, large multi-national organizations like Deutsche Telekom [90] or GoDaddy [104] that among other businesses also operate a certificate authority business. On the other hand, CAs can be small organizations like Buypass [2] whose main business is issuing of digital certificates. CAs represent trusted third parties that verify correctness of key bindings.

When a CA verifies a key binding, it issues a certificate to the requesting entity [66]. Requester can use certificate to prove that a certain public key indeed belongs to her because a CA digitally signs a certificate. Users verify the correctness of the certificate by running a digital signature validation algorithm using CAs public key. For this reason, every user has to be assured that a public key which signed a certificate actually belongs to a CA. This gives rise to a hierarchical tree-like architecture which characterizes CA-based PKI (Figure 2.1).

Although in the beginning, when public key cryptography was not used on such a large scale as today, there could be a single CA which issued certificates to entities, nowadays architecture of the currently most widely deployed PKI consists of several independent tree-like hierarchies [66]. There are several independent root CAs that mostly do not issue certificates directly to the users, but instead transfer this task to a number of intermediate CAs which represent intermediary nodes in a tree. Root CAs verify and observe the behavior of intermediate CAs. Intermediate CAs issue certificates to the users, and they are those from whom users request certificates when they need them. Entities that request the certificates correspond to the leafs in PKI hierarchy tree.

These approaches make CA-based infrastructure scalable as there can be many levels of intermediary CAs which delegate some of their tasks to a lower level. Still, there is a trade-off between depth of infrastructure and trust [66]. Problem arises from the fact that trust is not completely transitive. Trust decreases with the number of intermediary nodes not only because of the possibility of increase in

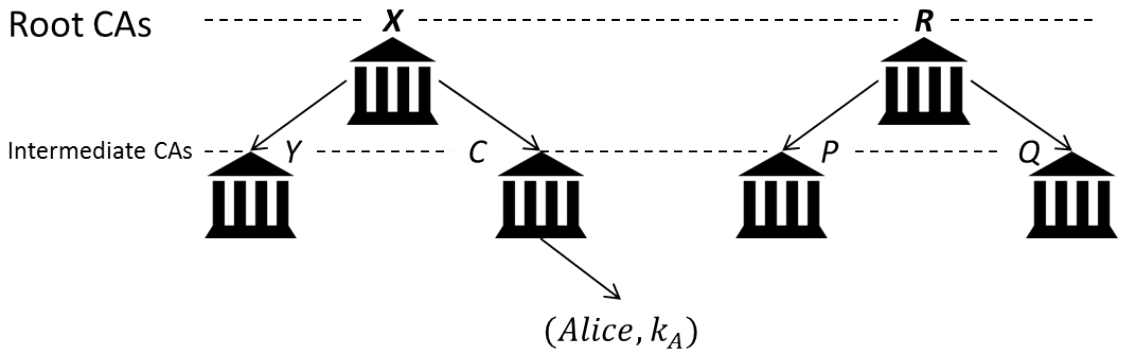

Figure 2.1: Hierarchical architecture with multiple roots used in CA-based PKI.

the number of malicious entities with the length of the chain, but also, even if all of the intermediary nodes were honest, the likelihood of a mistake by one of the intermediary entities increases. As a consequence of this fact, current web PKI has a flat infrastructure. Durumeric et al. find that in 98% of certificates there is only one intermediary CA between the CA that issued the certificate and the root CA [38].

When a client receives a certificate, she checks its validity by searching for a chain of trust from one of the root certificates to which she entirely trusts [14]. Such chains of trust are called certification paths and consist of a received user certificate, certificates of all intermediary CAs, and a root CA. After the client establishes the certification path, she checks signatures of all the certificates included in the chain up to the root certificate [14]. With reference to Figure 2.1, if Bob got a certificate from Alice, he can check the validity of it by following the certification path as follows:

$$cert_{\hat{k}_C}(Alice, k_A) \longleftarrow CA\text{-}cert_{\hat{k}_X}(C, k_C) \longleftarrow CA\text{-}cert_{\hat{k}_X}(X, k_X)$$

Here $cert_{\hat{k}_C}(Alice, k_A)$ represents a certificate that is signed by a CA C using a private key $\hat{k}_C$. This certificate binds Alice to a public key k_A. On the other hand *CA-cert* is a certificate issued to intermediary nodes which gives them rights to act as a CA. In this case $CA\text{-}cert_{\hat{k}_X}(C, k_C)$ denotes a CA certificate that is signed and issued by CA X. This certificate guarantees the correctness of (C, k_C) key binding and authorizes C to issue new certificates. $CA\text{-}cert_{\hat{k}_X}(X, k_X)$ is a self-signed root certificate.

CA-based PKI considers a case in which user's private key is accessed or stolen as well. If Alice's private key is compromised, she can revoke all of the certificates issued for the corresponding public key. Alice does this by informing the CA that issued the certificate about an intrusion. The CA publishes publicly and periodically a document called Certificate Revocation List (CRL) which holds the list of all revoked certificates that have not yet expired [14]. Therefore, when a CA receives Alice's request for revocation of her certificate, CA includes an identifier associated with her certificate into CRL. Consequently, when a client wants to connect to Alice's website, he first has to check whether Alice's certificate is included into CRL. If this is the case, the client knows that Alice's private key has been compromised and can distrust the certificate. However, CRLs can grow up to several MBs so that they strain connections. For example, after Heartbleed attack CRLs of the largest CAs grew rapidly. GlobalSign had to revoke 56,353 certificates in two days, whereas GoDaddy revoked 243,823 certificates in the next three months after the attack [39].

An alternative concept that does not suffer from this problem, but instead introduces a small bandwidth overhead on every connection establishment is an Online Certificate Status Protocol (OCSP) [100]. Client that uses OCSP requests status of the receiving certificate from an OCSP responder. OCSP responder reponds with *good*, *revoked*, or *unknown*, and signs the response message using its own private key if the requesting client trusts the key binding between an OCSP and its public key. If OCSP responder's key binding is not trusted, the response is signed using the private key of an issuing CA. Like certificates, OCSP responses have an expiration date as well. Weakness of OCSP is that it reveals user's browsing history to the OCSP server.

2.2.1 Digital Certificates

Certificates can be classified according to certificate issuance practice. CAs publish Certification Practice Statement (CPS) documents where they describe how they verify identities of the owners before issuing a certificate. Depending on the extensiveness of identity verification they offer different types of certificates. Most of the larger CAs tend to offer the following three main categories of certificates:

- **Domain Validated (DV) Certificate** This is the cheapest kind of certificates which has the lowest verification standards, but its issuance is often automatized which saves on time. According to Comodo CA, these certificates do not provide authentication and validation. They only provide encryption of communication between the client and the server [27]. Consequently, DV certificates should only be used in low risk environments. More specifically, firstly the requester generates a certificate signing request (CSR) with which she proves the possession of the private key [27]. CSR contains a public key, information about the requester, and the requester signs it using her private key [85]. If CSR is accepted, CA proves domain ownership by searching for contact information of the entity which registered the domain on domain name registrar and uses found contact information to communicate with the entity. The entity has to make some changes on the website to prove the ownership of domain. Examples of DV certificates are COMODO SSL, GlobalSign's DomainSSL, and free certificates issued by Let's Encrypt.
- **Organization Validated (OV) Certificate** These certificates offer a moderate security level. As in DV validated certificates, the requester generates a CSR and proves control over the domain in the same way. However, CA verifies contact information and the identity of the requester by, for instance, actually visiting the organization on the address in contact information, checking with the responsible government agency, or if it is about an individual, CA can verify an identity and address through an ID card issued by the government [27]. Examples of OV certificates are Comodo's InstantSSL, DigiCert's Standard SSL, and GlobalSign's OrganizationSSL.
- **Extended Validation (EV) Certificate** These certificates provide the highest level of security. Consequently, browsers represent EV certificates differently. On websites whose owners have undergone extended validation the location bar contains the name of an organization in green text instead of only "secure" [105]. When issuing EV certificates, CAs follow the guidelines set by CA/Browser Forum [55]. Guidelines prohibit issuance of EV certificates to a requester that does not qualify as a private organization, government, business, or non-commercial entity. Validity period of EV certificates is limited to 825 days. In addition to verifying a

proof of private key possession and domain ownership, CAs issuing EV certificates have to verify the subject organization information: requester's legal existence and identity, business presence and activity at a physical address, way to communicate with the requester, identity and authorization of persons involved in certificate issuance as well as the execution of actions in which these persons are involved.

2.2.2 Vulnerabilities

Centralization and Single-Point-of-Failure

Since global market share of Firefox and Chrome amounted to more than 80% in December 2018 [106], browser vendors have such a large influence on web PKI that they can decide which CAs are trusted by the web users. For this reason, in practice website owners would buy certificates only from the CAs whose root certificates are included into browser root certificate lists. This makes the current web PKI centralized.

In CA-based PKI every CA included in the root certificate list of the browser can issue certificates for any domain [38]. This makes every CA a single-point-of-failure because there is less than 150 organizations that have their certificates in root certificate lists of the browsers. What is more, most of the intermediate CAs can also issue certificates to any domain. This increases the number of trusted third parties to thousands. Durumeric et al. have found that in 2013 there were 1,832 CA certificates [38]. Admittedly, every CA included in root certificate lists presents a single-point-of-failure, thus smaller number of CAs included decreases the risk of misbehaving CAs. Nevertheless, compared to web-of-trust architecture on which PGP is based (Section 2.3 on page 28), small number of CAs reflects how centralized the whole system is.

To improve security, modern browsers and operating systems include a number of pre-installed trusted root certificates. Table 2.1 lists how many trusted root certificates are pre-installed in certain operating systems (OS) and browsers. Browser vendors take different approaches to their root certificates. On the one hand, Google Chrome relies on the underlying OS and uses the root certificate list of the OS on which it was installed. On the other hand, Firefox specifies 141 trusted certificates. Finally, while both OS vendors contain more root certificates than Firefox, number of trusted root certificates in Microsoft Windows stands out.

Not only there does the risk of coercion for CAs by the authorities exist, but governments of some countries themselves run their own CAs. Such CA entities are governments of countries like the Netherlands and Sweden as well as governmental agencies of some countries like the national mint of Spain. There are also large CAs like DigiCert that have many root certificates in all vendor lists. Since traditional PKI assumes that all these entities are trusted, further cases similar to the already mentioned Symantic could still be possible. Some OSs like macOS include different kinds of lists for root certificates. macOS differentiates between a list of (1) trusted root certificates, (2) always ask certificates, and (3) blocked certificates. This allows one line of protection by including CA certificates of compromised CAs into the list of blocked certificates. Blocked certificates are never trusted.

Because CAs mentioned in introduction showed significant malicious behavior, they can be found on blocked certificate lists of some vendors. Some CA certificates operated by Symantec and WoSign contained on the root certificates list of Windows OS are considered disabled, while simultaneously

Browser / OS	Number of trusted root certificates	Number of owners
Mozilla Firefox [22]	141	54
Google Chrome [94]	depends on OS	depends on OS
Microsoft Windows (10) [102]	292	123
iOS (11) [72]	174	79
macOS (10.14) [72]	174	79

Table 2.1: Table shows how many root certificates are included in browsers and OSs as well as the number of different certificate owners in these lists. As organization names are not listed in iOS and macOS CA root lists, we estimated their number from certificate names.

one WoSign's CA certificate could be found on the list of blocked certificates in macOS and iOS. As seen in the case of Symantec, browser vendors rarely, and only for significant misbehavior or mistakes, remove CAs from the root certificate list. Symantec was removed only after recurrent misbehavior that occurred between 2009 and 2017.

MitM

Intermediate CAs in the hierarchy can issue any sort of user certificate. Consequently, if an attacker can find a way to trick CA (by pretending to be Bob or taking control over CA) into issuing him a certificate which binds public key of his choosing, the attacker can then run a successful MitM attack against Alice, and the whole concept of privacy and security falls apart. If certificates are used on the web and Alice tries to connect to Bob's website using HTTPS protocol, attacker is able to intercept the message and pretend to be Bob by presenting the obtained unauthorized certificate. In this situation Alice would see a secure green bar in the browser although the attacker controls the connection. Steps of the MitM attack could look as follows:

1. Compromised CA issues an unauthorized certificate which claims that $(k_{att}, bob.com)$, where k_{att} is attacker's public key, is a valid binding although *bob.com* is actually bound to some other public key.
2. Alice tries to connect to *bob.com*.
3. Attacker intercepts messages between Alice and the web server and replaces the genuine certificate in the server's response with the unauthorized certificate received from compromised CA
4. Alice establishes an HTTPS connection with the attacker, but she believes that she communicates securely with *bob.com*.

Since in implementations clients might ignore CRL if they cannot obtain it, adversary that compromised Alice's private key can use MitM attack to include older CRLs or mount DoS attack on the CRL server to prevent clients from finding out that the certificate has been revoked. Same situation occurs in OCSP protocol [99]. If OCSP is used, an MitM attacker can undermine user's privacy because she can see which websites the user visits by intercepting the OCSP requests, or she can run a replay attack on the victim by injecting older OCSP responses. Chrome does not use CRL or OCSP, but instead includes CRLSet, a CRL maintained by Google [99].

When issuing certificates Let's Encrypt [61] uses ACME protocol which is vulnerable to MitM attack [7]. If Eve controls part of the network between domain holder Alice and ACME server and has access to DNS server, she could obtain a certificate which binds a public key of her own choosing to Alice's domain. At the time of writing there were almost 100 million active Let's Encrypt certificates [71].

2.3 PGP

PGP is an abbreviation for Pretty-Good-Privacy. It is a cryptographic software application developed by Philip R. Zimmerman that combines symmetric and asymmetric cryptography to get advantages of both approaches [32]. Its goal is to provide communication partners a way to communicate securely in a convenient fashion and to enable privacy in personal communication. PGP is nowadays mostly used for private e-mail communication. PGP is owned by Symantec since 2010, but there is also GnuPG, an open source alternative that implements OpenPGP standard defined in RFC 4880 [48].

There is a difference in the use of PGP term in literature. Firstly, PGP and open source version GnuPG can denote cryptographic application software that was first developed by Philip Zimmermann. Secondly, it is also often used in a more general sense of a Web of Trust, a type of PKI architecture. Thirdly, similarly to the synonymous usage of X.509 certificate standard and CA-based PKI, PGP often refers to OpenPGP certificate standard, an implementation of Web of Trust.

PGP uses a Web of Trust (WoT) PKI architecture. WoT is a decentralized architecture in which every user acts as a CA [66]. Alice uses her private key to sign public keys of other network participants to whom she trusts. Similarly to the CA-based PKI, the notion of trust chains is used. When Alice gets a message from Frank, she first checks whether she trusts Frank. If that is not the case, she then checks whether there exists a chain of trust between her and Frank. For example, if Alice trusts Bob, and Bob trusts Frank, then Alice can find a chain of trust to Frank, and she can accept the message (Figure 2.2).

When a user wants to revoke her key, she creates a revocation certificate and signs it using her private key. Similarly to OCSP approach to revocation in CA-based PKI, in PGP user posts signed revocation certificate to one of the public key servers [120].

As most PKI systems based on blockchain strive to make PKI decentralized to prevent censorship and single-point of failure problem, many of them either extend PGP, or base their work on it.

2.3.1 PGP Trust Model

PGP differs between the notions of validity and trust [115]. Alice considers Bob's key binding valid if she is assured that the public key belongs to Bob, whereas trust presents her belief that Bob is trustworthy when signing other key bindings.

Links between nodes in the graph of PGP implementation of Web of Trust are weighted. Namely, when Alice signs Bob's public key she not only attest that Bob's public key is valid, but also attaches a certain level of trust that she has in (Bob, public key) binding [48]. Hence, with every signature validity and trustworthiness of a key binding are specified. As a consequence of this, users can

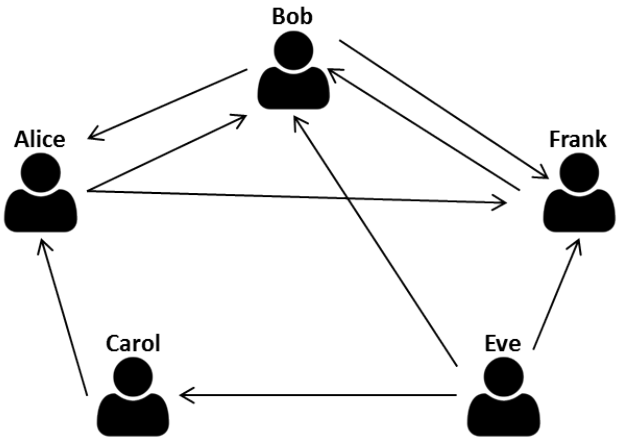

Figure 2.2: Web of Trust architecture. Arrows represent trust relationships.

decide how much they trust the receiver of their digital signature, and this can indicate an extent to which signer verified the receiver of the signature, which is similar to the certificate types in CA-based PKI.

Four levels of trust introduced by PGP are as follows [1]: (1) *unknown*, (2) *untrustworthy*, user has no trust in the owner of this public key, (3) *marginal*, user deems it mostly appropriate and (4) *full*. Apart from trust levels, PGP also uses the notion of validity levels which describe how much the user believes in the key bindings. Validity levels are: (1) *undefined*, (2) *marginal*, and (3) *complete*.

Nevertheless, since users have to decide themselves which validity and trust level are they going to pick, above-mentioned categorization has only an advisory role.

2.3.2 Usability, User Networks and Vulnerabilites

Usability proves to be a major weakness of PGP. In 1999, Whitten and Tygar did a user study and cognitive walk-through analysis on PGP 5.0 and came to conclusion that PGP is not suitable for use by most computer users [118]. Through cognitive walk-through they found numerous violations of usability best practices such as not easily accessible explanations of particular UI elements, having same icons for public and private key, not giving clues whether a button initializes local or remote call to key server, irreversible actions. More specifically, in a user study, twelve experienced email users represented participants. They had to send a signed and encrypted email to five different entities in the span of 1.5 h. Only three participants successfully signed and encrypted an email in the given time span. Similar usability studies that came to same conclusions were done in 2006 by Sheng et al. on PGP version 9.0 [101] and in 2016 by Ruoti et al. where they analyzed the usability of Mailvelop, a browser extension that allows users to encrypt email using PGP [95].

Network of users that share trust links between each other is called a strongly connected component or a strong set. Although mass adoption towards which its creator aimed never happened, PGP experienced steady growth since its inception. In January 1996, the largest strong set contained only 2,047 public keys [78]. Size of the strongest set grew steadily from less than 20,000 participants in 2003 to its peak at the end of 2018 when it amounted to more than 60,000 keys. As of December

10, 2018 the largest strong set consisted of somewhat less than 60,000 keys with an average mean shortest distance between the keys being between 6.2 and 6.3 [87]. It is also noticeable that there are users which have hundreds of signatures and these users usually have lower mean shortest distance to other keys. Lowest mean distance between keys in the largest strong set was 3.669.

Although PGPs usage of Web of Trust architecture makes it inherently decentralized, to propagate public keys PGP uses key servers. This introduces a single point of failure to the system in addition to the need for synchronization between servers which enlarges the attack surface. PGP key servers allow anyone to post key bindings, but they do not check whether the poster corresponds to the entity whose key binding she posts [115]. Key servers also contain a lot of obsolete public keys whose owners either lost corresponding private keys or do not use PGP anymore. Because of this, users cannot rely on the keys which they find on key servers. Moreover, users can also sometimes sign keys without completely verifying the identity and that way introduce malicious actors into a trust network. PGP is also susceptible to Sybil attack where the attacker creates many fictional identities [70]. Such an attacker could create large trust networks full of fictional users and that way try to trick honest users into singing her public keys.

2.4 CONIKS

CONIKS is a PKI that was developed as a response to the deficiencies of CA-based standard and its transparent log extensions [79]. In CONIKS there are four participating entities and four protocols through which entities interact with it. There is an identity provider authoritative for a certain namespace. Identity provider issues key bindings with identifiers in this namespace. Issued key bindings are stored as leaves in a key directory which is structured as a Merkle binary prefix tree. Identity provider repeatedly issues in time intervals a structure called Signed Tree Root (STR) (2.1). STR is a concatenation of t, a time interval number in which STR is published, t_{prev}, previous time interval number, $root_t$, root of the Merkle tree, $h(STR_{prev})$, colission-resistant hash of STR in the previous time interval, and P, condensed version of identity provider's security policy. STR is signed using identity provider's private key $\hat{k}_d$.

$$STR = T_{\hat{k}_d}(t||t_{prev}||root_t||h(STR_{prev})||P) \tag{2.1}$$

Users in CONIKS have a client software. Similarly to CT (Section 2.5.2 on page 32), client software monitors user's key binding using a lookup protocol by automatically downloading a proof of inclusion. Proof of inclusion contains all the intermediate nodes needed to compute STR, and they allow clients to verify the consistency of the key binding between different time intervals. If clients encounter a misbehavior, they can distribute their findings by gossiping with auditors which represent a third type of entity. Auditors can either be clients themselves or there can be third party auditors. Identity providers can offer auditing services too. Auditors observe a number of STR chains each of which is associated with a specific identity provider. When a new STR is published on one of the STR chains, auditors first check whether the STR signature is valid and then compare hash of the previous STR with $h(STR_{prev})$ contained in the current STR. Through gossip network auditors check whether the same key binding is presented to all the clients. CONIKS also offers two levels of privacy: (1) Alice's public key can only by accessed by someone who knows her name, (2) Alice can limit the visibility of her public keys to a certain set of users she trusts.

Particular feature of CONIKS is that there is no special key revocation operation. Since the public key in the last *STR* is considered to be currently active, revocation in CONIKS is done by updating the associated public key.

2.5 Extensions of CA-based PKI

There have been several proposals to improve PKI through extension of CA-based PKI based on X.509 standard. Several approaches look to monitor CA behavior through publishing of publicly and widely available data structures called transparency log. Examples of such proposals are AKI [64], ARPKI [8] and the most widely deployed such system is Certificate Transparency developed by Google [69].

2.5.1 Public Key Pinning

Public Key Pinning (PKP) is an HTTP extension with an existing standard described in RFC 7469 [45]. It enables users to follow trust on first use (TOFU) principle for encountered public keys on a certain website. Public key encountered in the first certificate presented by the web server during first HTTPS connection establishment is considered to be trustworthy. It is expected that every further HTTPS connection establishment in a certain time interval with the same domain will use the same public key.

Public key pinning extension introduces two new headers to HTTP protocol: *Public-Key-Pins* and *Public-Key-Pins-Report-Only*. While both headers make the client associate their content with a domain on first connection, when the client encounters unspecified public key *Public-Key-Pins-Report-Only* makes her only send reports to the URI specified in directive *report-uri*, whereas *Public-Key-Pins* also blocks the connection.

An example of a response header that the client might receive from the server can be seen in Listing 2.1. There two public keys are pinned using a sha256 hash of a Subject Public Key Info sections from the certificate; with *max-age* directive the period of the validity of the pin is set, and a URI to which reports of connection failures should be sent is specified.

The advantage of public key pinning is that it reduces vulnerability to MitM attacks if the first connection was not established while domain was undergoing an MitM attack. However, this comes at a cost. If an adversary succeeded to take over the first connection, adversary's public key will be pinned to the domain. Pinning keys instead of certificates allows CAs to issue new certificates that will be accepted by the client for a key that has already been pinned [45].

Public Key Pinning is included in both Firefox and Opera [59], unlike in Google Chrome where it was abandoned in 2018 in favor of Certificate Transparency [31].

Listing 2.1 Example of Public Key Pinning Header Response [45]

```
Public-Key-Pins: max-age=2592000;
         pin-sha256="E9CZ9INDbd+2eRQozYqqbQ2yXLVKB9+xcprMF+44U1g=";
         pin-sha256="LPJNul+wow4m6DsqxbninhsWHlwfp0JecwQzYpOLmCQ=";
         report-uri="http://example.com/pkp-report"
```

2.5.2 Certificate Transparency

Certificate Transparency (CT) is a transparency log system that extends CA-based PKI. It is created by Google and allows publishing of certificates to a public transparent log which makes early detection of unauthorized certificates and misbehaving CAs possible.

CT introduces three new entities to CA-based PKI [23].

- **Certificate Log** is an append-only log server. Parties can publish certificates or request proofs from the log. It uses an implementation of Merkle hash trees (Appendix A.3 on page 94) to provide two kinds of proofs: Merkle consistency proof and Merkle audit proof. Certificates represent the entries whose hashes are leaves of the tree. Merkle consistency proof is a set of nodes that prove (1) an earlier root node is now either one of the given intermediate nodes, or it can be computed from the given nodes and that (2) hash of newly issued certificates is in the log. Conversely, Merkle audit proof acts as a proof of inclusion by giving user a set of nodes that allow her to calculate the path from a leaf node, corresponding to the queried certificate, to the root node.
- **Monitor** is a server that keeps an eye on all of the certficate logs. It periodically checks for unauthorized modifications of log's content, certificates appended since the last check, and whether certificates with unusual properties (e.g. type of certificate can be different from expected) are present in the log. To do this a monitor usually copies the whole log. This process is made faster by making the monitor copy the log once and then periodically update it with newly appended certificates. For this reason monitors can also temporarily take place of unavailable logs. Monitors are usually run by CAs.
- **Auditor** is an application run in the browser that checks the presence of certificate in a log.

Less than 1,000 logs in the world is considered enough [23]. A possible reason for this is a trade-off between decentralization and monitoring difficulty because a monitor has to contact all logs. In February 12, 2019, there were 37 certificate logs operated by 10 different organizations included in Google Chrome [26]. Six logs were disqualified and of the remaining 31 qualified, Google and DigiCert controlled 9 and 15 CT logs respectively. This fact suggests that CT is at the moment highly centralized. Further gain in the use of CT was prompted by Google's decision to flag websites in Chrome as insecure if they do not publish their certificate on transparency log. After multiple postponements, ever since 2018, newly issued certificates are only accepted if they are posted to CT's logs [25]. In February 2019, there were 3,623,539,988 certificates registered in CT logs [112], and Stark et al. find that of 10 000 most popular websites on February 5, 2018 according to Alexa approximately 39% of them were CT-compliant [105].

CT does not include revocation, but it inspired other proposals such as Revocation Transparency (RT) [68] and Enhanced Certificate Transparency [96] which make revocation transparent as well.

2.5.3 DANE

DANE is an abbreviation for DNS-Based Authentification of Named Entities [37]. DANE adds a new record type called TLSA RR to DNS which allows users to link a public key or certificate with a domain name already in DNS. As TLSA relates to the resource record used, actual protocol is called DANE TLSA. DANE assumes that the DNS protocol is protected by DNSSEC. DANE gives users the possibility to link a certificate with a domain without specifying the issuing CA or even by only specifying the public key. This way domain owner can pin herself her public key with TLSA record. In this case, certificate usage TLSA parameter has to be set to DANE-EE, and certificate verification is done by only checking whether TLSA record associated with the domain exists. There are no certification chains. This however impacts compatibility with CT because in this situation certificate would not be published in CT's transparency log.

Adoption rates of DNSSEC validation on which DANE relies hovered between 10% and 20% since 2016 [114]. DANE solves two properties of Zooko's triangle: (1) human-meaningful names and (2) security (Appendix A.4 on page 95). Nevertheless, DANE is centralized because DNS root server represents the root of trust.

3 Blockchain

In this chapter we take a closer look at blockchain, the novel technology that shows properties which can improve the existing PKI. We look at its structure, the way it stores data and the way it allows decentralized consensus. In doing so, we orient on Bitcoin and Ethereum blockchain platforms. In the end we recognize the vulnerabilities that affect the blockchain because the security of a PKI system built on top of it depends on the security properties of the underlying blockchain.

Blockchain is a data structure that consists of a chain of blocks which contain transactions. It is an append-only immutable distributed storage which represents a global state on which all network participants agreed. Blockchain was developed to serve as a data structure underlying the cryptocurrency Bitcoin [83]. This was the first application of blockchain. It was used to build a decentralized electronic payment system that allowed users to hold and send a certain amount of digital tokens.

After Bitcoin, many other blockchain platforms appeared. Most of them were used to build alternative cryptocurrencies, but some platforms also looked into applications of blockchain other than the cryptocurrencies. Namecoin [84], first fork of Bitcoin, appeared in 2011. Although it concentrated on the introduction of DNS system on blockchain, Namecoin also introduced the possibility of storing arbitrary data on blockchain. Furthermore, Ethereum blockchain platform appeared in 2015. Ethereum significantly expanded possible applications of blockchain by introducing the possibility of distributed computation [20]. Small programs called smart contracts could be deployed on blockchain, and the network participants agreed on the result of the computation.

In general, there are two types of accounts in a blockchain network. Blockchain network participants that want to send transactions, i.e. change the global state, have to create a pair of public and private keys that serves as an account in the blockchain platform and allows users to sign transactions. In most blockchain platforms, such as Bitcoin, accounts are represented through their public keys, while in Ethereum an account is represented through a 20 bit address created by hashing the account's public key. Users and accounts are different entities. As public keys represent accounts, a single user can create many accounts. This opens every system that relies on accounts to a sybil attack. If blockchain supports smart contracts like Ethereum, there is another account type controlled by contract code called contract account [20]. User owned accounts in Ethereum are called 'externally owned accounts'.

Next, we take a look at the structure of the blockchain while putting emphasis on Bitcoin and Ethereum platforms as described in [83], [20], and [122].

3.1 Structure and Functionality

Blockchain contains a global state that is distributed between participants through a peer to peer network. Network participants therefore differentiate between a local state, which is represented by a blockchain stored on their node, and the single global state agreed upon by all participants. State in a blockchain platform depends on the functionalities offered by the platform. In Bitcoin, state consists of user's accounts which are linked to their Bitcoin balances. An account balance is implicitly built into a blockchain and can be read from transactions. It is calculated by summing up the values of all unspent transaction outputs associated with the account. Unspent transactions outputs are outputs of transactions that were not used as inputs to other transactions, i.e. coins that were received by an account, but not sent to another account. On the other hand, state in Ethereum is explicitly made up of all account states that look like this [122]:

$$(nonce, balance, storageRoot, codeHash) \tag{3.1}$$

In (3.1) *nonce* represents a number of transactions or contract deployments; *balance* is ether owned by the account; *storageRoot* is a hash of the root of Merkle patricia tree where account state is stored; and *codeHash* represents the hash of deployed contract code which controls contract account. *codeHash* is non-empty only in contract account states.

3.1.1 Transactions

Global state of the blockchain is changed through transactions. Each transaction is initiated by one of the network participant's accounts to whom it is linked by a signature. Sender uses her private key to sign a transaction. This gives transactions a non-repudiation property. Transactions in the blockchain usually transfer a certain amount of coins between two accounts. Alice, for example, wants to send Bob a certain amount of coins. In the course of this, it is taken into account that the transaction is valid. Transactions are considered valid only if Alice has enough coins on her account balance.

In addition to the coin transfer transactions, Ethereum has two other types of transactions related to its smart contract capability: smart contract creation and contract function call transactions. Network participant sends a contract creation transaction when she deploys a new contract to blockchain. This transaction consists of five elements [122]:

$$(init, nonce, gasPrice, gasLimit, to, value, signature) \tag{3.2}$$

In (3.2) *init* is a contract code which signifies type of transaction, while *nonce* is the number of transactions initiated by the sender. *signature* refers to the signature done by the sender of transaction using her private key, and *to* is empty in a contract creation transaction, but in general *to* represents the receiving account of the amount of ether specified in *value*. Other elements of transaction are specific to Ethereum by being related to the concept of gas.

Gas Since every contract execution requires a certain amount of computational and storage resources, in order to incentivize miners to use these resources, sender sends a certain amount of coins to the miner in form of a transaction fee. Transaction fees in Ethereum are represented in terms of gas which can be bought with ether. Price of gas included in a transaction is specified with *gasPrice*. Ethereum allows miners to specify *gasPrice* that they are willing to accept. Sender of transaction specifies the amount of gas to send with transaction in *gasLimit*. As every smart contract instruction has associated cost in gas [122], gas sent with a transaction is depleted during the execution of transaction. If there is enough gas, transaction is accepted and the remaining gas (specified in *gasLimit*) is returned to the sender. Conversely, sending an insufficient amount of gas results in an out of gas exception. This is also the prevention mechanism against infinite loops in smart contracts. Called function is executed until transaction runs out of gas [122].

In contrast, function call transaction (3.3) is sent to a blockchain when the network participant wants to call a function offered by some smart contract. In this case *to* represents address of the contract that is called, and *data* represents input data and implies the type of transaction. Other transaction elements are the same as in contract creation transaction (3.2).

$$(data, nonce, gasPrice, gasLimit, to, value, signature) \tag{3.3}$$

3.1.2 Blocks

In general, blocks consist of a block header and transactions included in them. Each block is identified by the hash of the block's content.

Transactions in a block are stored into a Merkle tree structure (Appendix A.3 on page 94) in which they represent the leaves of a tree. As transactions change the global state of the blockchain, order of their execution is important. Transactions are executed sequentially in the order in which they were stored into a block. Since miners decide in which order they are going to collect transactions this opens the possibility for an attack. Users can however also influence transaction order by offering to pay higher transaction fees because miners have an incentive to first collect transactions with higher fees.

Block header is the main part of the block. It contains hash of the block's content as well as the hash of previous block's content. Blocks are linked this way because every block points to a previous one through the hash of previous block's content. It gives rise to a chain of blocks and prevents malicious actors from tampering with the blockchain as every change to a block results in the change of its hash. For this reason, the used hash function has to be collision resistant. Bitcoin uses SHA-256, while Ethereum uses SHA-3. Bitcoin block header also contains a root node of the Merkle tree in which transactions are stored, whereas Ethereum's block header contains hashed root of the Merkle patricia tree that represents the state after all transactions are applied [122].

In block payload Bitcoin contains a collection of transactions stored with Merkle trees, while Ethereum blocks in addition to transactions also include Ethereum state. In Ethereum, for efficiency reasons, block's store only changes that were applied in this block [20].

Blockchain platforms differ by the number of transactions that could be included into one block. Some blockchains like Bitcoin have a set limit for the size of blocks. Although this means that each block can contain up to a certain number of transactions, up to this number miners can choose

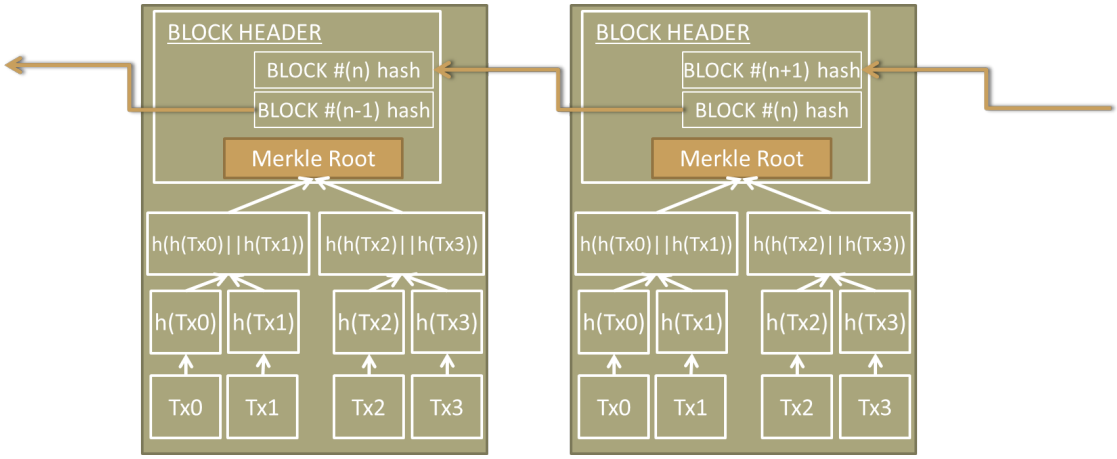

Figure 3.1: Example shows how are blocks stored in a full node in Bitcoin network [83]. Hash function is denoted by $h(\cdot)$, Txi represents transaction i and $||$ is a concatenation. Root Hash $= h(h(h(Tx0)||h(Tx1))||h(h(Tx2)||h(Tx3)))$.

themselves how many transactions they will include into a block making the number of transactions per block variable. The first block in the blockchain, known as the 'genesis block' is special. It is not mined and it does not have a hash that points to a parent block.

Blocks are issued in set time intervals. Time intervals depend on the blockchain platforms, e.g. in Ethereum blocks are issued every 15 seconds, while in Bitcoin this time interval is set to 10 minutes.

As of April 2019, size of Bitcoin blockchain is greater than 200 GB [29]. This growth was predicted by Nakamoto, therefore measures were taken to save disk space and make Bitcoin more portable. Transactions are hashed in a Merkle tree (Appendix A.3 on page 94) and the root of the tree is included in block header. As a consequence, there are two main kinds of nodes in Bitcoin network, full and lightweight nodes [83].

- **Full Node** contains local copy of the whole blockchain. It contains all transactions and with it the entire history of the blockchain (Figure 3.1). Blocks in full node contain block header with the whole Merkle tree containing all transactions.
- **Lightweight Node** does not contain the blockchain in its entirety in its local copy. Blocks in a lightweight node store only block headers (Figure 3.2). As block header makes only a small part of block's size, root of the Merkle tree contained in the block header is enough for it to be able to verify payments if it gets a set of transaction hashes that can be used as a proof of inclusion of transaction from a full node.

3.1.3 Consensus

To update the global state network participants execute consensus algorithm. Consensus algorithm allows users to decide which block extends the blockchain. Examples of consensus algorithms are Proof of Work (PoW) and Proof of Stake (PoS).

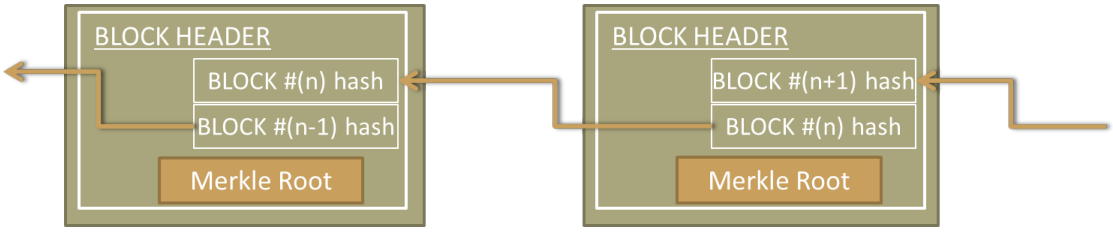

Figure 3.2: Example of blocks stored locally in a lightweight node in Bitcoin network [83].

Proof of Work (PoW) is a consensus algorithm used in Bitcoin. It forces network participants that want to extend the blockchain by adding a new block to it to solve computationally difficult problems. This activity is called mining and network participants that take part in the activity are called miners. They are rewarded for their work with transaction fees. In Bitcoin, miners have to solve a 'hash-puzzle', i.e. find a value v by hashing a block header which is lower than target T, for a block to be accepted to blockchain [116]. In (3.4), a miner can change *nonce* in order to make the value of hash lower than the target T; $hash_{prev}$ refers to the hash of the previous block, i.e. block before the currently created one; *merkleRoot* is the root of the Merkle tree made from transactions included into a block.

$$v = h(nonce||hash_{prev}||merkleRoot)) \leq T \tag{3.4}$$

Possibility of different values for T allows a blockchain to adjust to the computational power that is present in the blockchain network. Main disadvantage of PoW consensus procedure is that the network needs huge amounts of electrical energy. However, generation of electricity today is still mostly based on fossil fuels [60], and according to Taylor, Bitcoin mining had in 2018 requirements equivalent to 74% of total electricity generation in Austria [109].

Proof of Stake (PoS) presents an environmentally friendly alternative to PoW consensus algorithm. In PoS, a user that is to mine the next block is chosen according to the amount of coins, i.e. stake that she holds in a blockchain system [116]. Some of the established blockchain platforms like Ethereum want to transition to PoS. These platforms run both consensus algorithms at the same time. There is also a subtype of PoS called Delegated PoS where network participants vote to choose a certain limited number of delegates responsible for chain extension. Examples of blockchain platforms that use PoS consensus algorithm are Peercoin (firstly implemented PoS) and Emercoin.

Since multiple nodes can simultaneously publish blocks that contain differing transactions with correct solutions of the hash-puzzle, there can be more than one blockchain branch in the network. In Bitcoin this problem is solved by designating the longest chain as the chain that represents the global state. In Ethereum, on the other hand, the chain that represents the global state is chosen with a variant of GHOST protocol [20]. Chain of blocks with the most computation done on it is the one on which the network participants agree. Moreover, after a new block is appended to a blockchain, other network participants validate the block by comparing the hash value of the previous block with the hash of the previous node included in the block header, checking the proof of work, applying transactions, and checking their validity and whether they were already spent [20]. Updates to blockchain usually go through steps shown in (Figure 3.3).

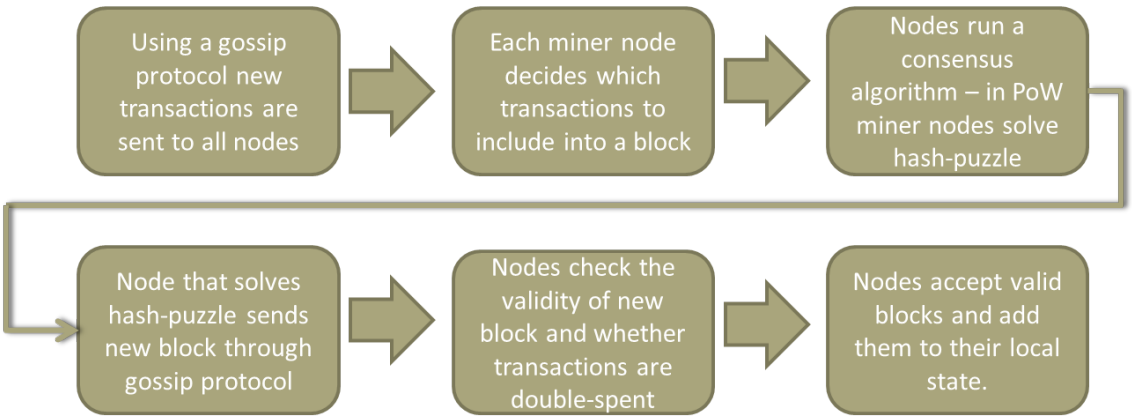

Figure 3.3: Steps through which nodes in the network go.

3.2 Blockchain Types

Blockchains can be classified according to a consensus algorithm they use as well as the openness of the blockchain network towards new participants. As PoW is widely used in blockchains, there is further classification depending on the function used as a proof-of-work. For example, Bitcoin and Namecoin use SHA-256, while Ethereum is based on Ethash proof-of-work function which uses SHA-3 hash function.

Depending on the openness of the network towards new participants, there are two main types of blockchain platforms: permissionless and permissioned blockchains.

- **Permissionless** — Blockchain network is open and public, anyone can join it [6]. Anyone can operate a node in the network or create blockchain accounts. Considering that users are not authenticated, permissionless blockchains offer a certain level of privacy to the users. Likewise, since access to the network is not limited, permissionless networks are usually much larger than permissioned. As the accounts in the network are not trusted, PoW consensus algorithm is needed to prevent sybil attacks. Both Bitcoin and Ethereum as well as most of the other largest blockchain platforms are permissionless.
- **Permissioned** — Blockchain network is private and there is an entity or a group of entities that restricts access to the network [6]. This central authority decides who can join and which rights will the user have in the network. Knowing that users have to be authenticated to determine whether they have access to the network, the permissioned blockchains offer lower level of anonymity than permissionless blockchains. Authentication increases trust in users which opens a possibility to use consensus algorithms other than PoW. Examples of permissioned blockchain platforms are Hyperledger Fabric and MultiChain.

3.3 Attacks on Blockchain

3.3.1 51% attack

Blockchain platforms based on PoW consensus algorithm are vulnerable to 51% attack. If there is an attacker whose computational power is higher than that of all honest nodes combined, then this attacker can generate new blocks faster than the rest of the network. Attackers branch will grow faster than an honest branch and it will, at some point in time, become the longest branch. This allows an attacker to achieve double spending.

In a double spending attack, immediately after an attacker sends Bob a certain amount of coins for a service, attacker Eve starts mining in secret a chain of blocks, but instead of including the transaction to Bob, Eve includes into one of the blocks another transaction where she sends coins to an account that belongs to her or her accomplice [36]. When the block containing transaction to Bob is published, Bob sends goods to Eve, but as Eve has more than 50% of computational power she can mine blocks faster and after some time her secret chain of blocks gets longer than the accepted chain. Eve then reveals her secret chain, but because her chain is now the longest, it is considered to represent the actual global state and is accepted by the network. Other nodes would consider it to be a branch with the most PoW and continue mining on it.

In order to mitigate 51% attack, blockchains often have incentives in form of transaction fees that motivate network participants to behave honestly. Thus, even if an attacker has more than 51% of computational power, she also has an option to behave honestly and collect transaction fees. This increases opportunity cost of mounting an attack for adversary. If an attacker can acquire more coins by using the computing power available to her for mining rather than by mounting 51% attack on a blockchain, then an adversary would not attack blockchain if her only motivation is profit. Furthermore, to protect against 51% attack Bitcoin defers confirmation of transactions. Transactions are considered confirmed only after six succeeding blocks are accepted to the blockchain [28].

Arbitrary changes by an attacker are still not possible because nodes will not accept invalid blocks and payees will not accept invalid transactions as payments. For example, as transactions are signed by the sender, the attacker cannot create unauthorized transactions between accounts that do not belong to her. Lightweight nodes are even more vulnerable to 51% attack. The attacker can completely fool them as they cannot verify the transactions themselves [83]. To mitigate the attack on lightweight nodes, Nakamoto proposes that honest full nodes inform other nodes when they detect invalid blocks.

51% attack is the most serious problem for new blockchain platforms that do not have a lot of hashing power to protect against it. In 2018 there were numerous cases of successful 51% attacks on smaller blockchains such as Monacoin, Bitcoin Gold, Zencash etc. [13] However, even blockchains that are protected with more computational power are vulnerable to 51% attacks. In January 2019, there was a successful 51% attack on Ethereum Classic where attackers double spent $1.1 million [30].

Nowadays, mounting a successful 51% attack could be done by renting computing power on crypto-mining marketplace like NiceHash [67]. This further lowers the requirements needed for an attack. Potential attackers do not have to buy needed hardware, but can instead simply rent it. According to *crypto51.app* website [88], the cost to arrange 51% attack for 1 hour on Ethereum Classic using

Blockchain	Market Cap	Algorithm	1h Attack Cost	NiceHash-able
Bitcoin	$90.25 B	SHA-256	$360,190	0%
Ethereum	$17.48 B	Ethash	$91,393	4%
Bitcoin Cash	$4.99 B	SHA-256	$20,729	1%
Litecoin	$4.83 B	Scrypt	$47,226	4%
Monero	$1.11 B	CryptoNightR	$5,369	5%
Ethereum Classic	$697.51 M	Ethash	$6,502	60%
Bitcoin Gold	$283.42 M	Zhash	$1,395	11%
Metaverse ETP	$44.86 M	Ethash	$329	1,187%
Expanse	$1.46 M	Ethash	$60	6,546%

Table 3.1: Costs to mount 51% attack on certain blockchains [88]. NiceHash-able column refers to the percentage of the hash power needed for an attack available for rent on NiceHash [67].

hardware rented from NiceHash would currently amount to only $6,502 (Table 3.1). To put this amount into perspective, we mention that the market cap of Ethereum Classic is $697.51 M at the time of writing.

Costs calculated are based on the assumption that NiceHash has enough hash power available to mount an attack. For larger blockchains like Bitcoin, Litecoin and Ethereum these values are under 5%. However, if NiceHash had enough hash power to attack Bitcoin and Ethereum, the potential cost would amount to only $360,190 and $91,393 respectively. This would make even the largest blockchains vulnerable to 51% attacks from state actors and large organizations. Thankfully, potential attacker would have to buy computing power difference that is not available for rent. This increases significantly the price to attack some blockchains. To exemplify, Bonneau estimates that to buy enough hardware to mount an attack on Bitcoin or Ethereum, one would need approximately $1.5 billion [16].

3.3.2 Concentration of mining power

One of the main advantages of blockchains for PKI is that they are decentralized. In practice, however, on many of the blockchains, the largest mining pools, communities of miners that combine their hashing power in order to increase the rewards they receive, control a significant part of hashing power and make decentralization of blockchains questionable.

According to [17], two largest mining pools in Ethereum, Ethermine and SparkPool_3, mined 48.82% of blocks in the seven days leading up to April 17, 2019 (Figure 3.5), whereas in the case of Bitcoin four largest mining pools have mined more than 50% of blocks in the same time period (Figure 3.4). Moreover, Ethermine and SparkPool_3 mined 49.51% of Ethereum blocks in March 2019 [17]. What is more, Antpool and *BTC.COM*, Bitcoin's largest mining pools are operated by the

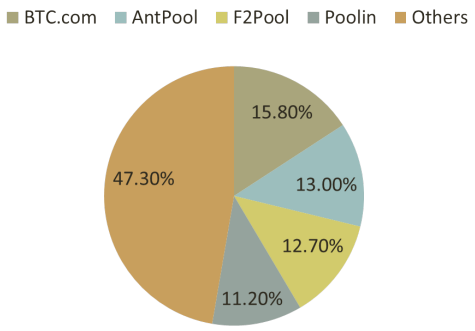

Figure 3.4: Bitcoin blocks mined in the week leading up to April 17, 2019 [18].

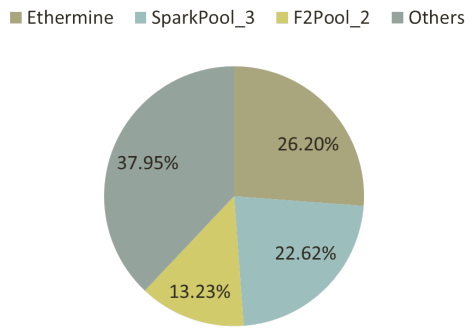

Figure 3.5: Ethereum blocks mined in the week leading up to April 17, 2019 [17].

same company, an Application-specific integrated circuit (ASIC) producer Bitmain [11]. In 2018 during some weeks Antpool and *BTC.COM* mined more than 40% of blocks [76]. Blockchains with less hashing power such as Namecoin are even more susceptible to this problem and were shown to be very centralized. Ali et al. [3] noticed that since 2014 a single mining pool consistently controlled more than 51% of Namecoin's hashing power.

In 2018 Gencer et al. [53] also noticed that Bitcoin's nodes are geographically less spread out than Ethereum's nodes and that their available bandwidth speeds are higher. They posit that the reason for this is that Bitcoin's nodes can more frequently be found in data centers.

3.3.3 Eclipse Attack

Eclipse attack presents a vulnerability of peer-to-peer networks. In eclipse attack an adversary tries to isolate the user from the rest of the blockchain network. As the networks of permissionless blockchains such as Bitcoin are open for anyone to join, this fact also opens the door for an adversary. Furthermore, nodes in Bitcoin network are identified only by their IP addresses, hence users do not know the actual entities controlling the node [57].

When an adversary mounts an eclipse attack, she separates Bob from the rest of the blockchain network by making the victim interact only with the nodes that belong to the adversary. Nodes needed for an attack can be created by running a Sybil attack and creating multiple identities. As users check whether they have the same global view as other users by communicating with other network nodes, this allows the attacker to manipulate Bob's view on the global state. Attacker can use this fact to run a double-spending attack on Bob. Attacker mines some blocks in which she includes a transaction that says she sent some coins to Bob. She then shows Bob a view on blockchain containing these blocks that she mined that contain transactions where she sent coins to him, while showing the other nodes a view where coins were spent on something else [57]. As Bob thinks he has received the coins, he might consider the transaction fulfilled and send bought articles to the attacker.

In [57] Heilman et al. showed an eclipse attack on the Bitcoin blockchain network. They succeeded in running a double-spending attack. Ethereum peer to peer network has been shown to be vulnerable to eclipse attack as well [75].

Possibility of eclipse attack can be decreased by not allowing all the connections to be incoming, by making it obligatory for a certain number of outgoing connections to be initiated by the node. Introduction of anchor addresses that are not affected by reboots and to which the node always connects if they are available can also decrease the chance of eclipse attacks [57]. Finally, anomaly detection is another technique that can be used against eclipse attack.

3.4 Blockchain's PKI

Knowing that accounts in blockchain are represented with their public key, and they digitally sign transactions using asymmetric cryptography, there is a need for PKI in blockchain itself. Since users in permissionless blockchains are not authenticated, one party can create many accounts. Through the need to calculate PoW every time one wants to extend the blockchain, blockchain's PKI links a certain number of accounts to an entity with computational power behind them [52]. Accounts controlled by an entity are represented by her public keys, and blockchain's PKI implicitly binds these public keys to a single entity which controls them.

Garay et al. presented a bootstrapped backbone protocol based on the core of the Bitcoin on which they developed a PKI. They showed that there is no need for an arbitrary genesis block, but that it is possible for multiple entities to converge on a single blockchain [52]. Their proposal is resistant to an adversial precomputation attack where malicious entities try to mine a certain number of blocks in advance and get them into the final single blockchain.

Garay et al. also prove that their protocol generates a PKI with an honest majority. Their protocol allows solving two hash puzzles at the same time and introduces transactions that have to be mined before they are published to a blockchain network and are able to be included into new blocks. One hash puzzle, considered solved if found hash is lower than target difficulty, is used for mining new blocks, but authors also introduce another hash puzzle which is considered solved if found hash is higher than some target difficulty. Two PoW schemes are independent. Newly included hash puzzle is used for mining transactions, and the number of transactions generated by any account is bounded by the computational power behind it.

In order to generate PKI, an entity inserts public keys in transactions together with a hash of the earliest block that is part of the blockchain prefix common to all entities agreed upon in the bootstrapping phase. After a local blockchain reaches a certain length, all transactions that contain a hash pointing to the above-mentioned block contain the same public keys as the local blockchains of the other honest entities, and more than 50% of these public keys were generated by honest entities because of the assumption that honest miners hold majority of the computational power. Finally, pseudo-anonymous level of privacy is provided by this system, and the established PKI can be used to initialize PoS in a way that more than 50% of coins is owned by honest parties.

4 PKI Systems based on Blockchains

Below we present PKIs that are based on blockchains. PKI systems that are mentioned in discussion are shown in Table 4.1 on the following page. In order to describe and highlight the differences between them as well as to enable easier comparison between the existing systems, we highlight the relationships between the systems and look into ways in which we can classify them. Thereafter, we present some of the PKI systems that belong to defined categories.

4.1 Classification

We classify PKI systems according to the two main concepts that distinguish them. PKI systems can be classified according to (1) a blockchain on which they are built and (2) a PKI model which they extend or improve. In Table 4.1 and Figure 4.1, 21 PKI systems are classified according to these criteria. Some of the PKI systems are not based on any of the traditional PKI models, instead they combine DNS with PKI. We call these systems authoritative systems (Section 4.5 on page 55).

It is noticable that there are clusters of systems that are based on a certain PKI and built on top of a certain blockchain. We identify four such clusters:

- **Authoritative systems based on Namecoin:** Certcoin, PB-PKI, and Blockstack ID
- **CA-based systems on Ethereum:** BlockPKI, IKP, and Yakubov et al.
- **PGP-based systems on Ethereum:** SCPKI, BlockPGP
- **CONIKS-based systems on Bitcoin:** Catena, Conifer

Moreover, we also note the existence of proposals such as Authcoin, BARS, and CertLedger which either have their own custom blockchains or do not specify the underlying blockchain platform. In Authcoin it is assumed that the system has its own independent blockchain, while BARS uses three custom blockchains that are adapted for their use cases. CertLedger requires the underlying blockchain to support smart contracts, but is otherwise blockchain agnostic.

Bitcoin blockchain is the source of almost all of the systems not built on Ethereum because other blockchains either grew as forks from Bitcoin or are a fork of a blockchain based on Bitcoin, as is the case with Peercoin and Emercoin. Peercoin is a fork of Bitcoin that implements a hybrid PoW and PoS consensus algorithm, and Emercoin is based on Peercoin. This means that most of the PKI systems stem from Bitcoin.

Another way to classify PKI systems can be by their application. Some PKI systems are intended to be used in a web environment where domains are bound to corresponding public keys, whereas others are more general and bind any name to key bindings. Binding user's identities with their public keys is usually application of PGP-based systems, however, there is an exception in form of X509Cloud

	Blockchain	PKI	Incentive Mechanism	Privacy	Implementation	Year
Certcoin [51]	Namecoin	/	✓		✓	2014
Nidaba [97]	Bitcoin	/	✓			2014
Wilson and Ateniese [120]	Bitcoin	PGP	✓		✓	2015
EmerSSL [43]	Emercoin	/	✓	✓	✓	2015
EthIKS [15]	Ethereum	CONIKS	✓	✓	✓	2016
Authcoin [70]	unspecified	PGP			✓	2016
PB-PKI [4]	Namecoin	/	✓	✓		2016
Blockstack [3]	Bitcoin	/	✓		✓	2016
Catena [111]	Bitcoin	CONIKS	✓	✓	✓	2017
Cecoin [91]	Bitcoin	CA-based	✓		✓	2017
SCPKI [9]	Ethereum	PGP	✓		✓	2017
IKP [77]	Ethereum	CA-based	✓	✓	✓	2017
X509Cloud [110]	Multichain	CA-based				2017
Conifer [35]	Bitcoin	CONIKS	✓	✓	✓	2018
BARS [73]	3 custom	CA-based	✓	✓	✓	2018
CertLedger [65]	unspecified	CA-based	✓			2018
BlockPGP [124]	Ethereum	PGP			✓	2018
BlockPKI [40]	Ethereum	CA-based	✓		✓	2018
Yakubov et al. [123]	Ethereum	CA-based	✓	✓	✓	2018
Ghazal [80]	Ethereum	/	✓	✓	✓	2018

Table 4.1: Under privacy we consider systems that provide unlinkability between identities and public keys or allow entities to access data associated with a name only if they know what they are looking for. System is considered to have an incentive mechanism if there are rewards for activities such as the extension of blockchain.

which is CA-based, but is concentrated on distribution of client certificates. Furthermore, BARS and PB-PKI are to be used in Internet of Things (IoT) or vehicular networks and concentrate on privacy in a way that makes public keys unlinkable with entities behind them. Examples of systems that are intended to mainly be used in web PKI are: CertLedger, Certcoin, Ghazal, and Cecoin. Although authoritative systems in general could allow registration of different namespaces and names, which makes them flexible in their application, PKI systems are usually implemented within a single namespace, e.g. Blockstack ID with namespace *u/* on Namecoin, Ghazal with *.ghazal*. CONIKS-based systems are similar in this regard as they also allow entities called identity providers to control namespaces for which they are authoritative.

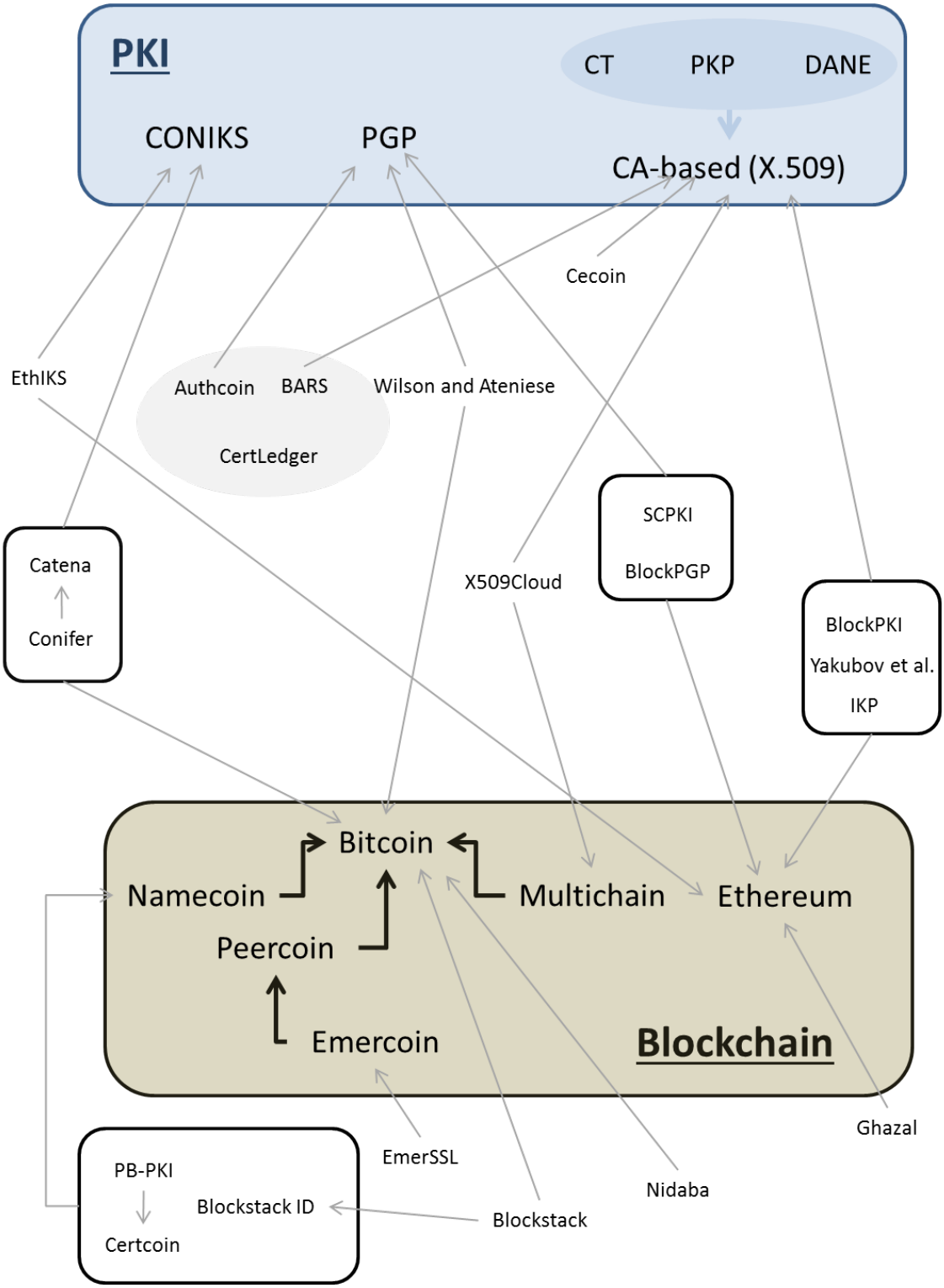

Figure 4.1: Shows on which blockchain is a specific PKI system based and to which traditional PKI it is linked. Links between blockchain-based systems are also shown if some system is inspired by, or extends another one.

4.2 Blockchain-based PKIs related to CA-based PKI

4.2.1 X509Cloud [110]

X509Cloud is built on top of a permissioned blockchain MultiChain. It aims to increase adoption of client user certificates by decreasing the cost of issuing certificates and making revocation more secure by obsoleting the need for CRLs. It stores certificates in a blockchain and allows both revocation of issued certificates and look up of stored certificates. Furthermore, by giving institutions and organizations the role of CAs, X509Cloud allows employees to use certificates issued by their organization's CA for internal access. Cloud service could allow users to log in to websites using one-time passwords instead of having to use a username and password.

To link users with their public keys, X509Cloud uses certificates based on an X.509 standard. Certificates are issued by organizations such as universities, governmental institutions, or companies. These organizations issue certificates to their employees, and by doing that, act as trusted third parties or intermediate CAs.

Client certificates stored in a blockchain are mapped to users' email addresses. This allows querying for certificates using email addresses of the clients. Authors claim that the use of MultiChain results in a lookup time of $O(n * log(n))$.

Operations supported by X509Cloud are as follows:

- **Certificate request / issuance** An employee of an organization can request issuance of a certificate after passing two-factor authentication. This operation relies on existence of accounts related to the organization such as for example university accounts. To request a certificate, Alice uses a smartphone application. She generates a key pair in the application and like traditional CA-based PKI, Alice sends CSR to the organization's server. Difference is, however, that organizational CA has information about Alice. Organization verifies the signature and the information contained in CSR. If the verification is successful, the organization creates an X.509 certificate, posts it to the blockchain, and sends it to the client.
- **Certificate revocation** Since last certificate issued for a user on the blockchain is considered to be valid, certificate is revoked by issuing another certificate for the same user. This makes two-factor authentification a requirement for revocation as well. Authors give an example where an employee leaves an organization. In this case organization issues a new certificate for a soon to be ex-employee in which it gives her lower rights. Hence, employee's previous certificate is considered revoked, and she is blocked from internal access.
- **Certificate lookup** Web services can query blockchain for stored certificates. When Alice visits a website, web service can query Alice's current client certificate on the blockchain. Query returns a certificate with a matching username that is found in the block with the highest block number, i.e. most recently mined block containing the username.
- **X509Cloud Service Registration** Alice registers for the service using her email address.

X509Cloud is the only system which uses MultiChain, a private permissioned blockchain that is a fork of Bitcoin [54]. Similarly to Namecoin, MultiChain allows network participants to send arbitrary data in transactions. What is more, amount of data which can be stored can be set in a configuration file that is defined on creation of a new blockchain. To decrease the possibility of malicious miner taking over, MultiChain uses the concept of 'mining diversity'.

Mining diversity parameter can be between 0 and 1. After Alice mines a block, mining diversity value affects the proportion of other network participants that are allowed to mine blocks before Alice can again mine the block [54]. While setting mining diversity to 1 offers the highest security, this makes Alice wait until all of the other users mine blocks. Due to the possible inactivity of network participants, such a high value could negatively affect the usability and scalability of the platform because there might be long periods of time when blocks are not mined. For this reason, creators of MultiChain propose 0.75 as the value for mining diversity that achieves balance between security and usability [54].

4.2.2 IKP [77]

IKP is an abbreviation of Instant Karma PKI. One of its main goals is introduction of instant and automatic reactions to detected CA misbehavior, and introduction of rewards and punishment that disincentivize misbehavior. IKP uses X.509 certificates to attest key bindings and aims to improve CA-based PKI by employing several smart contracts built on top of the Ethereum platform.

Five entities present in IKP are: (1) CAs, (2) domain owners, (3) IKP contract, (4) Clients, and (5) Detectors. As in traditional CA-based PKI model, CAs issue digital certificates to domain owners, and clients obtain these certificates from domain owners during HTTPS connection establishment. In IKP, reaction policy (RP) is a novel product, which offers automatic reaction to misbehavior of CAs, that domain owners can buy from CAs. Furthermore, transactions between a CA and domain owner as well as other entities are mediated by the main entity, an IKP contract. Additionally, another novel entity, a detector, sends reports in transactions to an IKP contract when it spots a rogue certificate.

To further describe the interactions between entities and the structure of IKP, we first look at two novel constructs introduced in IKP:

- **Domain Certficate Policy (DCP)** When a domain owner wants to register a domain in the system, she defines a DCP which she registers using an IKP contract operation. DCP consists of preset fields which describe how a certificate authorized by a domain should look like. For example, DCP can restrict which CAs can issue a certificate for a domain. Likewise, DCP includes an Ethereum address of the 'check contract' which can be run to check the authorization status of the certificate. Moreover, included in DCP is the payout Ethereum address to which a remuneration is paid if a CA misbehaves. To mitigate the case where a domain owner loses access to her payout address, DCP also includes the possibility of specifying an update address which is used to update DCP, and can be different from payout address. When a DCP is updated its *Version Number* field changes as well.
- **Reaction Policy (RP)** RP is an additional product offered by the CAs which domains can buy independently of buying a certificate. An RP contract is always linked to a DCP, and a domain can buy multiple RPs from different CAs. RP specifies the reaction to detection

of an unauthorized certificate and the rewards and charges that arise from such misbehavior. Link between DCP and RP is only valid if both contracts have the same version numbers. RP always specifies a 'reaction contract' which is run when a misbehavior is detected. Every RP is valid only in the specified time interval and can prompt the execution of reaction contract only once because reaction contracts can send ether by issuing rewards and punishments. Similarly to certificates, an RP can be revoked if the issuing CA is found to be malicious.

DCP and RP contracts are associated with check and reaction contract types. Check contract is a smart contract defined by a domain owner who associates it with a DCP contract. Check contract receives a certificate as input and returns its authorization status in boolean form. Authors give an example where a domain owner defines a set of CAs which can issue certificates for a domain, and a check contract checks whether the CA of an input certificate is in this set. Furthermore, check contracts can be reused and it is possible to call multiple other check contracts from one such contract. This allows a domain owner to include into her own contract already pre-defined checks. Since check contracts can demand that a certificate is issued using a number of the existing extensions to CA-based PKI such as CT and public key pinning, these checks can be reused in multiple check contracts.

In IKP contract there is a list of RPs sorted by their expiration time so that on detection of unauthorized certificates an RP with the earliest expiration date can be executed. An RP contract is associated with a reaction contract which is executed on misbehavior detection. Reaction contract has a *trigger* method that is called by an IKP contract if check contract confirms that a reported misbehavior occurred. Depending on the measures taken, reaction contracts can be subdivided into 'client-facing contracts' and 'payout contracts'. Client-facing contracts fire events which are logged on the blockchain and signify that the certificate is considered revoked. An interested party can query blockchain for entries related to a certain certificate. If this party finds events fired by a client facing certificate, it implies that the queried certificate is invalid. Authors give an example where a browser extension could query blockchain and show a warning message to a user if it encounters such an event.

IKP contract is the main component of IKP system and only one such contract is deployed on Ethereum network. These are the main operations through which entities interact with an IKP contract:

- **CA Registration** CA registers to IKP by sending an identifier, information about itself to an IKP contract as well as proving the ownership of the chosen identifier. Ownership is proved with an existing certification chain to a chosen root CA in a traditional CA-based PKI model. On registration, the CA can also transact a certain amount of ether to an IKP contract where it is associated with the CA and can affect its reputation.
- **DCP Registration** Domain owner includes into registration transaction a domain name which she wants to link to a DCP together with a corresponding proof of ownership. Included ownership is proven using DNSSEC or traditional CA-based PKI. On the one hand, registrants prove ownership by presenting a signature chain to the root DNS server. On the other hand, registrants can prove ownership through usage of signatures signed using private keys whose corresponding public keys are linked to the domain through certificates issued by at least three CAs which are already registered in IKP. Domain owner can also update DCP, but if the change affects *check contract* field, version number of the DCP changes and version numbers of all RPs associated with that DCP have to be updated.

- **Certificate / RP Purchase** IKP contract acts as an intermediary between a domain and CA for agreed price and validity period. At first a domain transacts the amount to pay and the hash of certificate or RP to an IKP contract. CA responds by sending a certificate or RP to IKP contract. In the case of an RP, CA also sends funds intended for a payout reaction contract. IKP contract then checks whether the hash of the certificate or RP matches the hash value sent by the domain. If the hash values match, ether is forwarded to the CA and either the domain receives the certificate, or ether from CA is forwarded to a reaction contract. At the same time issued RP is added to a list of RPs that is associated with the specified domain.
- **Reporting Misbehavior** Detector reports misbehavior by transacting a report fee with a certificate deemed to be unauthorized. However, since this opens a detector to an attack where a miner rewrites contents of transaction and pretends to be a detector, misbehavior reporting is done in two phases separated by a certain number of blocks. In the first phase a detector sends a hash $h(C||s)$ where C is a certificate and s a secret bitstring. Only after waiting a few blocks does the detector send a reported certificate C and secret bitstring s to IKP contract. IKP contract hashes C and s and compares it with an earlier value. If values match, IKP contract runs a *check* method associated with the DCP of a domain specified in the certificate. If reported certificate is found to be unauthorized, associated reaction contract is automatically triggered.
- **Client Lookup** Client looks up a certificate associated with a domain on the blockchain. It also checks whether there is an event on the blockchain which implies that the certificate is unauthorized.

Incentives in form of rewards built into IKP system have a secondary influence which is to induce the adoption of the system. The awards given out when a CA misbehavior is detected make it attractive to play the role of the detector in IKP, whereas the remuneration paid out in the same case stimulates adoption of IKP under the domain owners. CAs also get an opportunity to offer another product in form of RPs. Moreover, by analyzing prices and warranties offered by the largest CAs, authors found the highest risk of issuing unauthorized certificates to be 8.5%. For this reason, authors propose storing 10% of the specified payout in the payout reaction contract.

IKP decreases time during which a domain is vulnerable if an unauthorized certificate exists. Rewards to detectors incentivize them to quickly report unauthorized certificates and RPs allow automatic reaction in such cases. Compared to traditional CA-based model, domains do not depend anymore on CAs to manually revoke the certificates. Moreover, domains can terminate RPs as well if they change their opinion on trustfulness of the issuing CA.

Authors also look at collusion attacks where a CA tries to profit by manipulating an IKP reward system through collusion with another entity such as detector. This attack as well as the one where detectors spam the system with false reports are disincentivized through fee structure. Authors show that it is possible to set rewards and fees in such a way that the collusion attack is not profitable if the certificate gets reported. Attacker can profit if an unauthorized certificate is not reported, but incentives given to detectors decrease the probability of this case. Finally, a CA, which does not register in IKP, can profit through collusion with an RP issuer. Nevertheless, profit acquired this way is lower than it would be if the CA did not misbehave.

4.3 Blockchain-based PKIs related to PGP

4.3.1 SCPKI [9]

SCPKI is a PKI system based on a smart contract built for Ethereum blockchain. Similarly to PGP, SCPKI uses WoT architecture, but compared to other PKI systems, SCPKI does not store certificates into blockchain. It instead allows users to sign arbitrary data and store it in form of 'attribute' structures into blockchain. There is no limit to a number of attributes, hence every user can store an arbitrary number of attributes on the blockchain.

In SCPKI system there are two main parts: (1) client, which allows the user to interact with smart contract and InterPlanetary File System (IPFS) (Appendix A.5 on page 95), and (2) smart contract. There are two versions of the smart contract. Both of them are implemented in Solidity, a programming language used for writing contracts on Ethereum platform. The goal of smart contract is to provide functions that a user can use to interact with the blockchain, i.e. modify the global state of blockchain. In the smart contract, an entity is identified by her Ethereum address. An entity can post attributes that can contain arbitrary data to blockchain. Attributes are represented as structs in Solidity and are linked to its owner entity through her Ethereum address. *Attribute* struct contains an Ethereum address of its owner as a struct member.

There are three functions that an SCPKI contract supports:

- **Addition of Attributes** A new *Attribute* struct (Listing 4.1) is created which is linked with an entity that owns it through her Ethereum address. Caller specifies attribute's type and posted data together with its hash is added to the created struct. Besides, SCPKI gives users the possibility to state whether they have proofs for attributes that they add. For example, if a user adds a public key to data, she can also include a signature of her Ethereum address created using the corresponding private key. After the addition of attributes, an event *AttributeAdded* is fired. This event writes information about attribute's owner and contents to blockchain.
- **Signing of Attributes** By calling *signAttribute()* function Alice can sign an attribute that belongs to Bob. In order to link an entity with signatures she made, every signature structure contains an Ethereum address of the signer. After an attribute is signed, contract fires an event *AttributeSigned*. This event posts to blockchain information such as who signed who's attribute and signature's expiry date.
- **Revocation of Signatures** Since a user can later find out that some entity whose attribute she already signed is incorrect, there is also a *revokeSignature()* function which allows the user to revoke a signature. This function adds the revoked signature to an array of revoked signatures. The array contains only a list of revoked signature IDs and serves as a CRL. Revocation of the signature fires an event that logs revocation id and an id of the revoked signature.

Distinctive features of SCPKI are that it stores attributes instead of certificates, includes two smart contract versions, and uses IPFS. SCPKI's use of attributes allows higher granularity in signing of certificates. This extends the possibilities of PKI because users can sign only those pieces of information which they know to be correct. Although attributes can be arbitrarily chosen, if SCPKI is to serve as PKI, Alice should post her public key and information that identifies her. A light version of SCPKI smart contract has lower gas costs and storage requirements. Blockchain is only

Listing 4.1 Attribute struct in SCPKI full smart contract [9].

```
struct Attribute {
  address owner;
  string attributeType;
  bool has_proof;
  bytes32 identifier;
  string data;
  string datahash;
}
```

used for storing signer's addresses, attribute ids, and signature ids. Instead of storing attributes, signatures, and revocations of users on blockchain, light version stores these on IPFS (Appendix A.5 on page 95) and only hash of the stored data is posted on blockchain.

4.3.2 BlockPGP [124]

BlockPGP is a PKI system based on Ethereum that aims to improve PGP. In contrast to other Ethereum-based systems, BlockPGP does not intend to deploy contracts on the main Ethereum network, but instead has its own independent network. Using a separate Ethereum blockchain instance allows authors to make it permissioned. This allows them to run less computationally intensive consensus algorithm which makes the system more scalable. Instead of using PoW, BlockPGP uses proof of authority which gives only some of the nodes in the network the right to mine new blocks. This partly centralizes the system, but decreases the size of the blockchain that full nodes have to store. That being said, there are no transaction fees in BlockPGP, as no computational power is needed to mine blocks.

Key bindings in the system are represented with standardized OpenPGP certificates. Authors use comment field to add additional information to OpenPGP certificate. They include Ethereum account address of the certificate owner and expect signers to verify this address together with the rest of information about the certificate owner. Furthermore, while PGP key servers allow anyone to post key bindings that relate to any identity, BlockPGP restricts access to this functionality. Certificate data can only be updated by an administrator or certificate owner.

Authors implemented key server prototype that consists of a command line interface to functionalities offered by a single smart contract. BlockPGP allows users to post new certificates to blockchain, sign certificates of other users, and revoke made signatures, or owned certificates. Certificate signatures are, however, not immediately posted openly to blockchain but have to be accepted by the certificate owner. BlockPGP does not support trust levels.

4.4 Blockchain-based PKIs related to CONIKS

4.4.1 Catena [111]

Catena is a 'witnessing scheme' that uses Bitcoin. It allows creation of publicly verifiable logs and can be used to secure different systems. Catena logs are non-equivocating, i.e. they are prevented from showing different content to different users. Authors modified CONIKS to include STRs in Catena log.

Catena logs are represented by a chain of Catena transactions. Log owners have a Bitcoin account which they use to sign Catena transaction. Log is identified by 'a genesis transaction', the first transaction in log's chain of transactions that has to contain enough coins to pay miners for extension of the transaction chain. Like Blockstack and the system of Wilson and Ateniese, Catena transaction uses an unspendable output marked with OP_RETURN to log state changes or operations. Nevertheless, unlike these systems, Catena transactions also have a continuation output that is used as a first input to the next Catena transaction in the log's chain which gives rise to a transaction chain. Logs can also refund a transaction chain through other inputs. Mining fees associated with the extension of transaction chain represent costs of running a Catena log.

Auditors verify logs by going through the transaction chain. They start at the genesis transaction and check whether transactions are signed and linked correctly. Similarly to a six block rule in Bitcoin, in order to avoid forks in a blockchain, transaction chain waits a certain number of blocks before considering transaction confirmed. Furthermore, auditors can run lightweight nodes with only a few tens of MBs, and they have to download less than 1 KB, block headers and a proof of inclusion into Merkle tree, every time a new block is mined. For efficiency reasons headers are not promulgated in Bitcoin's peer-to-peer network but in a separate Header Relay Network (HRN).

If an adversary steals private key from the log owner, she can break log's transaction chain. Log owner can, however, start a new transaction chain. Like blockchain network, HRN is also vulnerable to eclipse attacks.

4.4.2 Conifer [35]

Conifer, based on CONIKS and Catena, allows identity providers to run operation logs which consists of an 'operation forest' and transaction chain. Operation forest is a data structure separate from blockchain that contains 'operation trees' ordered by their time of issuance.

Operation tree is a balanced Merkle binary search tree which only stores changes that affected the key bindings in one time period between STRs. Nodes in a tree contain an (1) index to which a name is mapped, (2) operations, e.g. registrations and name updates, and (3) hashes of child nodes. Stored operations consist of a public key bound to a private key allowed to sign the next operation, signatures signed by cryptographic identities specified in the previous operation, and information linked to the name. Operation tree supports proofs of name inclusion and absence. To provide non-equivocation, hashes of root nodes of operation trees in an operation forest are stored in Catena transactions in a transaction chain.

Registration of new names is not covered in Conifer. Identity providers can define registration procedures. To lookup a key binding in Conifer, client first gets the transaction chain. Client then gets operations applied to the name and for all time periods it obtains proofs of inclusion or absence of operations in operation trees which it uses to validate proofs with respect to the root nodes in transaction chain. Finally, client validates links between operations, i.e. checks whether the signature in an operation is signed using cryptographic identity specified in the preceding operation.

4.5 Authoritative Systems

Namecoin was the first fork of Bitcoin and the first so-called 'altcoin' [84]. Like Bitcoin, it uses PoW consensus algorithm, and new blocks are mined in the same way as they are mined in Bitcoin. Namecoin's main goal was to decentralize a domain-name system and make it resistant to any sort of censorship. Nevertheless, as Namecoin introduces the possibility to associate a certain amount of arbitrary data with names registered on the blockchain, it is suitable for implementation of PKI. Namecoin was the first naming system that offered a solution to Zooko's triangle (Appendix A.4 on page 95).

Operations supported by Namecoin platform are: (1) domain name registration, done in two steps, after Alice sends a hash of the chosen name, she has to wait for 12 blocks before registering the name, (2) update of data associated with a domain, and (3) domain name renewal. Domain names registered with Namecoin use *.bit* as a Top-level Domain (TLD). Since domain names and data related to them are stored on the blockchain, Namecoin acts as an authoritative name server for domains.

There are two kinds of fees for operations. Fee for registration of domain names (0.01 NMC coins) and transaction fees that apply to all supported operations [84]. Registration fee is currently static although developers expressed an intention to make it dynamic in the future. Block size is slightly lower than in Bitcoin and is between 500 kB and 1 MB.

The reason why there are several PKI systems based on Namecoin is that it appeared shortly after Bitcoin (at time when there were no other platforms such as Ethereum) and offered improvements that made it suitable for PKI. For example, Namecoin allows registration of new namespaces and a feature that was used in Blockstack ID system [3]. In contrast to Bitcoin which only allows storing 80 bytes in special transactions, Namecoin allows network participants to store 520 bytes of data associated with a certain name into blockchain. In addition, name uniqueness is enforced in Namecoin [84].

If public keys are associated with names in name-value pairs, Namecoin itself could act as a PKI system. In this case there is no need for certificates to be issued by a trusted third party like CA because there is no need to check the proof of name ownership. Entity that registers a domain adds a public key as the value in the name-value pair which proves that the same entity owns both domain and the associated pieces of information. We call systems that exhibit this property authoritative systems.

Although size limitation of values that can be associated with names makes Namecoin unable to store large arbitrary data, however, one could store a content hash into and explain how and where to find the content (e.g. content could be stored on IPFS). Further, Namecoin uses the concept of

'merged mining' to increase its resistance to 51% attacks. As Namecoin is mined in the same way as Bitcoin, merged mining allows Bitcoin miners to mine Namecoin blocks at no extra cost. This allows Namecoin to tap into the computational power that protects Bitcoin network.

4.5.1 Certcoin [51]

Certcoin is an authoritative PKI system based on Namecoin. Special feature of Certcoin is that it allows users to recover from a stolen private key. As Certcoin is based on Namecoin, it uses merged mining and it is limited to *.bit* domains.

To increase robustness of the system and allow key recovery, domain owners in Certcoin are associated with two key pairs:

- **Online key pair** – stored on web server, used for interaction with the client
- **Offline key pair** – used in online public key update, private key recovery, and public key revocation. It is called 'offline' because private key should be kept offline for maximum security.

Certcoin supports all operations necessary in every PKI system:

- **Domain registration** User chooses online and offline key pairs that are linked to her domain. To facilitate key recovery, for every created key pair user has to choose three trusted Certcoin accounts among whom the private key is shared.
- **Public key update** Any public key (online or offline) associated with a domain can be updated using this function. Domain owner sends a transaction containing:

  $$(d, T((d, k_{new}), \hat{k}_{old}))$$

  where d is a domain, k_{new} is a new public key, $\hat{k}_{old}$ is an old public key, and (d, k_{new}) is signed using the old private key. This way a user extends a chain of public keys that is stored on blockchain, thus the history of public key changes is stored transparently.
- **Public key lookup** At first it is checked that domain is not registered multiple times. As every public key change is signed with the previous private key, user can follow the public key chain and retrieve the last key in the chain. This allows users to verify the correctness of the public key at the same time. Moreover, the user also checks the validity of zero knowledge proof that proves the domain owner possesses the corresponding private key.
- **Private key recovery** Private key can be recovered using two of three trusted accounts designated on domain registration, or using offline public keys.
- **Key revocation** As only the last key in key chain is considered valid, public key is automatically revoked when it is updated. There is also the possibility to use offline key to sign a statement which invalidates all keys after a certain point in the public key chain.

Existence of public key chains allows domain owners in Certcoin to mitigate attacks where adversaries access or steal their private keys. Alice's private key is considered accessed if an adversary gets to know Alice's key, but Alice can still access it as well. In contrast, private key is stolen if an adversary gets to know Alice's key, and Alice cannot access it anymore. Honest user's can recover

in all situations except if both of their online and offline keys are stolen. Situations where one of the keys is stolen, or where both online and offline keys are accessed can be mitigated if user uses older keys to sign statements that revoke newer keys.

In order to support lightweight nodes, authors propose to store a single Merkle hash tree (Appendix A.3 on page 94) on the blockchain. Domain-key bindings together with an optional expiration date are stored into Merkle tree, and users running lightweight nodes have the ability to lookup public keys associated with domain names.

4.5.2 Blockstack [3]

Blockstack is the only observed blockchain-based PKI system that is in production. Blockstack is an open and permissionless layered system in which the lowest layer is Bitcoin blockchain. In 2016 Blockstack had 55,000 users.

Before Blockstack, Ali et al. operated another PKI system on Namecoin called Blockstack ID. In Blockstack ID, 33,000 name-value pairs were maintained, and 200,000 transactions were made. Blockstack ID defined a new namespace *u/* which became the most actively used namespace on Namecoin. Since Namecoin can associate only a limited amount of data with a name, in order to store arbitrary amounts of data, Blockstack ID used a linked list of name-value pairs. However, while they were maintaining Blockstack ID on Namecoin, authors discovered several vulnerabilities in Namecoin blockchain. Therefore, they decided to migrate their PKI system to Bitcoin.

In Blockstack, state changes, i.e. operations are stored on blockchain. They are ordered by time of addition. Likewise, registration of names and key bindings is separated from their data storage and distribution. Data in Blockstack is stored in zone files (4 KB per file) and external storage such as IPFS. Blockstack is built in layers on the blockchain. This gives Blockstack greater flexibility because it does not bind it with the underlying blockchain platform. There are four layers:

- **Blockchain layer** State changes, i.e operations are stored in ordered transactions.
- **Virtualchain layer** Blockstack operations and logic that decides which operations are accepted are defined in this layer. In virtualchain layer names are bound with a corresponding zone file hash.
- **Routing layer** DNS zone files used for routing are present on this layer. Distributed hash table network is used for zone file storage, but a separate routing layer also allows the use of different storage providers.
- **Storage layer** Name-value pairs stored in storage layer are signed by name owners. There are two modes that can be used at the same time: mutable storage and immutable storage. Mutable storage mode contains URI that links to actual data which contains name owner's signature. It is called mutable because changing data does not change the hash of the zone file, and consequently there is no need for blockchain transactions. Conversely, immutable storage, apart from URI, also includes data hash in the zone file. Changing data stored that way requires blockchain transactions because the zone file has to be modified as well.

Routing and storage layer do not have to be trusted because zone file hashes are stored on the previous layer.

Operations supported in Blockstack are: (1) name / namespace registration, (2) update of data associated with a name, (3) name transfer, (4) name renewal, and (5) name revocation. Names are registered in the same way as in Namecoin, on a 'first come, first served basis'. However, names are unavailable for registration for a certain amount of time after revocation. Blockstack also supports different namespaces which can define distinct pricing functions for names and registration costs. For example, developers also had to pay $10,000 to register *.id* namespace used in Blockstack's PKI system.

5 Discussion

Firstly, we notice that some PKI systems were developed by extending some previously existing systems (Figure 5.1). For example, Certcoin scheme was based on Namecoin system and it further developed its PKI features. Afterwards, Axon and Goldsmith extended Certcoin system by adding privacy to it. This kind of development avoids reinventing already existing features and helps in making systems compatible with each other, thus prompting faster adoption of new PKI systems.

Some PKI systems are not based on any of the traditional PKI models. It is a novel kind of PKI without trusted third parties that is enabled by the blockchain technology. We call such systems authoritative systems because as they combine DNS with PKI, they are authoritative for the domain names. Examples of such systems are Certcoin, Blockstack, and Ghazal. They present a solution to the risk #5 raised by Ellison and Schneier where they questioned the authority of CAs on the content (domain name and identity information) of certificates they issue [42].

Introduction of blockchain to PKIs does not resolve problems associated with verification of identity information. In CA-based PKI, issuance of OV and EV certificates still demands more extensive background checks of certificate requesters. Authoritative systems are also only authortitative for domain but not identity data associated with it. Furthermore, PGP-based systems make assumptions that signers know entities whose public keys they sign, while particularity of X509Cloud is that it's organizational CAs are actually authoritative for identity information which they check.

In comparison with CA-based PKIs, blockchain-based PKIs that do not use CAs can offer extremely low-cost certificates. For instance, in authoritative systems users only have to pay for registration of the name-value pairs. Considering the exchange rate on April 20, 2018, this can be as low as 0.02 USD in EmerSSL for a five year long registration of a certificate [43].

Security Single point of failure problem associated with every CA in CA-based PKI and possible equivocation by service providers in CONIKS is mitigated in blockchain-based systems. On the one hand, systems based on CA-based PKI and CONIKS use transparency and decentralized features of blockchain to make them accountable and decrease trust into CAs and identity providers. On the other hand, PGP-based systems decentralize the key server, the only centralized part of PGP. Finally, authoritative systems discard the notion of trusted third parties.

Security of authoritative systems is positively influenced by the lack of need to check the proof of domain ownership. As a consequence of this, the attack surface is restricted, and on PKI system level authoritative systems are more resistant to MitM attacks because compared to CA-based systems there is less communication between entities during CA issuance. Further, as there are no CAs, an MitM cannot present an unauthorized certificate to the client as the client can check which public key is bound to the domain on the blockchain. Nevertheless, on blockchain level they are still

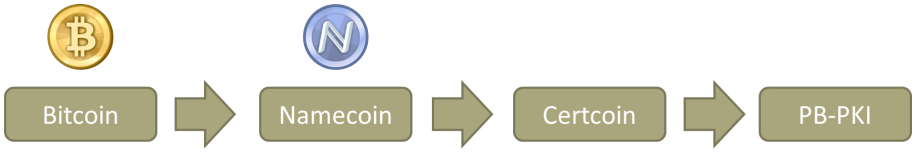

Figure 5.1: Evolution of PKI systems.

vulnerable to eclipse attacks, and although they implement preventive measures such as expiration of domains and possibility of transferring names between entitites, first come first served approach to name registration makes them open to domain squatting [84].

Blockchain-based systems that extend CA-based PKI decrease trust in CAs by forcing CAs to store issued certificates transparently on blockchain, employing incentive mechanisms, or creating automatized reactions to CAs misbehavior. Furthermore, domain owners can revoke certificates themselves without having to inform potentially malicious CAs to revoke certificates for them. Revocations can be logged on blockchain and that way blockchain-based systems avoid problems associated with CRL and OCSP. Revocation is visible to other blockchain network participants immediately when the block containing the revocation transaction is mined; thus, situations such as the one where a client has an old CRL are not possible. Clients running full nodes can check the certificate for revocation on their local blockchain without having to contact a third party as in OCSP. Privacy of the clients is therefore preserved.

Although the use of incentive mechanisms can disincentivize profit-oriented attackers, such mechanisms offer no protection against state actors who do not care about losses. Even if the PKI systems did develop structures against such attacks, the underlying blockchain protocol is vulnerable to them. Such adversaries could mount 51% attack on blockchain and double-spend transactions that were accepted by the participating parties (Table 3.1 on page 42).

Security of PKIs on blockchain can also be negatively affected by bugs in the implementation of blockchains. For instance, Namecoin suffered from a bug where a domain could be updated by any user, not only domain's owner [3]. Besides, systems that are based on smart contracts also have to be aware of the possible existence of bugs and loopholes. In one case an attacker exploited a loophole in the DAO (Distributed Autonomous Organization) smart contract on Ethereum platform and in the course of this stole 70 million USD [47]. This implies that bugs should be considered on all layers in the system.

Blockchain-based systems which allow registration and account creation can suffer from sybil attacks if they do not authenticate the users. In Bitcoin itself this is solved through PoW where the use of computational power for state changing actions makes the creation of multiple accounts with the goal of influencing the blockchain pointless [52]. In CONIKS-based systems the identity provider runs the log on blockchain, therefore sybil nodes have no influence on it, while PGP-based systems' use of trust mitigates sybil attack.

Despite the fact that the use of blockchain decreases the amount of communication between the client and server, issuance of DV certificates is still vulnerable to strong MitM attacks where an attacker can manipulate DNS entries. Some systems such as BlockPKI and Authcoin propose automated issuance of certificates using ACME. While automation increases the usability of systems, it also restricts them to DV certificates and opens them to MitM attack [7].

Adoption Authoritative systems can only vouch for key bindings of only one subset of possible domains, only for namespaces for which their naming system is authoritative. In PKIs built on Namecoin for example, these are domains that end on *.bit*, in Ghazal only domains that end on *.ghazal*, and Blockstack is authoritative only for domains that end on *.id*. This presents a great hurdle for adoption of these systems as mentioned namespaces are not yet supported by major browsers. All websites that do not use these domain names would have to register domains on these systems. Nevertheless, client certificates issued in EmerSSL are already supported for authentication in the main Emercoin mining pool [43].

PKIs based on blockchains have differing goals. We can subdivide them into two groups depending on their complexity and extent of their modifications.

On the one hand, there are larger systems with many entities and constructs. Such systems usually extend traditional hierarchical PKI. Although they make large changes, they are still based on users' trust in CAs. IKP and X509Cloud are examples of such systems. As these systems extend the most widely used PKI model, they would presumably have a lower hurdle for adoption. Nonetheless, although CT makes smaller changes to CA-based PKI system compared to PKIs based on blockchain, it still took more than five years until CT, which was backed by Google, developer of the most widely used browser, got adopted. Even so, the reason why CT was so successful is that Google made it mandatory for trusted certificates. Another factor in adoption of blockchain-based systems might be the popularity of blockchain platform. Consequently, PKIs based on Bitcoin and Ethereum might have a higher chances of adoption.

On the other hand, authoritative systems that combine a naming system with PKI can be much simpler than other systems. As there are no CAs, number of entities drops significantly. There is also no need for an extensive incentive system since there are no logs that contain certificates, and also no detectors that report noticed misbehavior of the CAs. Only incentive that is implemented in such PKIs is in the form of transaction fees through which miners are incentivized to extend the blockchain. Simplicity of these systems is also illustrated in the fact that most of such systems are implemented on top of Namecoin because they do not need programmability. However, programmability as seen in Ghazal makes the system less rigid and allows for easier transfer of domains between users. Basing Ghazal on Ethereum allows authors to implement an auction for transferring of domains between users. The fact of the matter is that although these systems seem to be conceptually simpler, they often require adaptation of the underlying blockchain.

As the currently most widely used PKI system is CA-based, this implies that adoption of any blockchain-based PKI system depends on its compatibility with CA-based PKI and the convenience of structures which it offers for the shift from the CA-based PKI. Although not as important as CA-based PKI, association with PGP might also positively influence adoption rate. Since PGP is predominantly used by power users, attracting these users might be easier for blockchain-based systems. Nevertheless, low usability of PGP might hinder adoption rates of blockchain-based systems that extend PGP (Section 2.3.2 on page 29).

Of all the systems in Figure 4.1 on page 47, only EmerSSL and Blockstack are used in practice, while Blockstack ID was active in the past.

Blockchain	Type	Consensus	Hash Power	Size	Avg. Block Frequency	Transactions Avg. per Minute
Bitcoin	permissionless	PoW	49,511,248 TH/s	247.56 GB	608 s	4.391
Ethereum	permissionless	PoW	146.773 TH/s	209 GB	13.3 s	8.094
Namecoin	permissionless	PoW	35,900,994 TH/s	5.61 GB	586 s	0.003
Multichain	permissioned	round robin with mining diversity	/	/	adjustable	1055.4 [81]
Peercoin	permissionless	PoW & PoS	18,734 TH/s	0.8554 GB	520 s	0.003
Emercoin	permissionless	PoW & PoS	11,632,214 TH/s	0.298 GB	532 s	0.005

Table 5.1: Blockchains underlying PKI systems. Data as of April, 2019. Sources: [29], [54], [12]

Importance of intended environment Although privacy-aware systems where links between entities and public keys are hidden might be useful for some applications, there are use cases where they are not needed or even counterproductive. For instance, linking domain names with their public keys should not be private because users would then be unable to use these keys to establish TLS connections as they would not know the corresponding domain. Furthermore, if we look at user certificates, where users are entities bound to some public key, we notice that hiding key bindings would contradict the goal of user certificates which is to make user authentication easier. Although PB-PKI could still be used in these scenarios because it gives entities the possibility of disclosing their identities, additional functionality that it contributes would not be used.

Incentive mechanisms Incentive mechanism decreases the risk of 51% attacks by increasing computational power that safeguards the blockchain. However, there are relatively few systems that make significant use of the cryptocurrencies inherently included in blockchain platforms. Incentive mechanism of most of the systems comes down to transaction fees that are paid to miners to add transactions containing key binding data to blockchain. There are even systems that do not have an incentive mechanism at all, e.g. X509Cloud, BlockPGP, and Authcoin. X509Cloud assumes that having their certificates stored in a blockchain that serves as the only certificate log would motivate CAs to extend the blockchain, whereas BlockPGP is built on an independent Ethereum network which uses proof-of-authority consensus mechanism that does not make use of computational power. As extending the blockchain in BlockPGP does not demand computational costs, the need for transaction fees is reduced. Authcoin, however, intends to adopt incentive mechanism in the future.

A major advantage of IKP system is its extensive use of the underlying cryptocurrency to incentivize entities to manifest good behavior. Through its fee and reward structure IKP makes attacks costly for adversaries while at the same time rewards good behavior. What is more, it also builds its reward mechanism in such way that it does not allow entities that try to cheat the system through collusion to profit from it. Conversely, although in CT public log in collusion with malicious CA can accept unauthorized certificates, CT has no automatic reactions or punishments for collusion attacks [111].

Underlying blockchains We find that most of the proposals are based on only a few blockchain platforms: Bitcoin, Ethereum, Namecoin, MultiChain, and Emercoin. It is noticeable that excluding MultiChain, all other blockchains are permissionless (Table 5.1). The reason for this might be that most of the PKI system proposals have as a goal to either extend or replace the current CA-based PKI model. As this means that a large number of participants might take part, permissionless blockchains seem to be more suitable. Still, BlockPGP and X509Cloud use permissioned blockchains. Further, the main goal of building a PKI on blockchain is to provide decentralization, but permissioned blockchains introduce a level of centralization. However, permissioned blockchains allow the use of more environmentally friendly consensus algorithms. For example, BlockPGP's version of Ethereum uses proof-of-authority, and MultiChain underlying X509Cloud uses round robin scheme with mining diversity. As permissionless blockchains do not authenticate network participants, systems built on this type of blockchain are therefore prone to sybil attacks because a single user can create multiple accounts. Finally, all of the blockchain-based PKI systems make an assumption that blockchain is secure although there are vulnerabilities that affect the blockchain (Section 3.3 on page 41).

As Bitcoin-based systems can only store 80 bytes per transaction, systems such as Nidaba, Catena, Conifer, Cecoin and Blockstack only use blockchain as an immutable log on which they log operations taken in their systems, and if they issue certificates, they store them in a separate construct. Especially in the case of Blockstack whose layered design gives it flexibility which allows it to switch the underlying blockchain. On the contrary, system by Wilson and Ateniese stores OpenPGP certificates in multiple transactions that have a Bitcoin address specified in the certificate as an input.

Nevertheless, authors of Blockstack find that even storing only operations does not make Bitcoin blockchain scale to millions users [3], Blockstack even had to be throttled in order to not overwhelm Bitcoin network. Further, we note that according to Table 5.1 only Ethereum has higher average number of transactions than Bitcoin, but if Blockstack had to be throttled on Bitcoin network, we assume that every blockchain network would be overwhelmed if PKI system would be used by millions of users. As a possible solution, Ali et al. propose to log multiple operations with a single transaction [3]. Using separate verifiable structures to store operations like in Conifer and storing only current hashes of those structures on blockchain might also offer a solution to scalability problem. Finally, using permissioned blockchains like MultiChain can raise throughput of the network, but comes at the price of a certain level of centralization, e.g. in MultiChain miner of the genesis block acts as an administrator of the network [54].

The higher the blockchain transaction throughput, the faster the growth of the blockchain. According to Table 5.1, sizes of the most popular blockchains already exceed 200 GB, and with the current transaction throughput Bitcoin grew on average 1.2 GB monthly while Ethereum grew 3.37 GB [35]. For this reason, most of the blockchain-based PKI systems support lookup of key bindings with lightweight nodes.

Another method that PKI systems use to mitigate the scalability problem of blockchains is to use multiple blockchains. For example, apart from loging operations on Bitcoin blockchain, Nidaba stores certificates on a custom blockchains. Every Nidaba blockchain holds a limited number of certificates. When this limit is reached, two new blockchains for certificate storage are created. Conversely, BARS uses three custom blockchains which allow it to adapt them to the needed functionality and data stored on them (e.g. certificate blockchain provides efficient proof of inclusion, whereas blockchain with revocations provides efficient proof of absence).

Centralization The main reason why PKI systems use blockchain is decentralization. However, there are some clues that question how decentralized are blockchains actually in practice. We assume that the high hash power that protects Namecoin in Table 5.1 is a result of the use of merged mining. Such a high hash power should make users feel secure, but Ali et al. find that this in fact offers a false sense of security. A single mining pool at times controlled up to 75% of computational power [3]. Moreover, Judmayer et al. analyzed merged mining and concluded that merged mining increases the risk of centralization in blockchains [62].

There are some that claim that high levels of centralization of the mining power in hands of mining pools in blockchains do not affect blockchain's centralization level because mining pools do not act as a single entity, but depend on interests of individual miners within the pool. Sui et al. find that over 4,000 individual miners would have to collude to launch 51% attack on Ethereum [107]. On the other hand, Gencer et al. claim that individual miners can hardly detect censorship by pool operators [53]. The fact that in Bitcoin an ASIC producer Bitmain held more than 40% of the hashing power indicates an even larger problem because this producer also has an effect on the supply of the hashing power. In 2018 Bitmain held 75% of the market share in Bitcoin ASIC market [49].

In traditional PGP new signatures are often created on PGP signing parties where users meet each other in person and exchange their identity documents. However, there are many situations where such documents could have been obtained fraudulently [50]. Moreover, as identity documents are issued by governments, one could say that signatures acquired this way are centralized because there are a few roots of trust in the whole system all of which present a single point of failure (similar to hierarchical architecture with several root CAs). Although this questions the decentralization of WoT architecture, not relying on identity documents would assume that users know each other which would seemingly present too high of a barrier for new users to enter the web of trust. Nevertheless, the idea of WoT architecture agrees with the notion of six degrees of separation which posits that each person is linked to every other person through at most six other persons [5]. What is more, Backstrom et al. find that in 2012 Facebook users were on average separated by less than four friends which also implies that the problem of trust transitivity should not significantly impact WoT.

Not all PKIs on blockchain are completely decentralized. As CA-based systems still use CAs they keep a level of centralization, but otherwise blocks on blockchain are mined by a diverse group. Nevertheless, there are systems such as Cecoin, X509Cloud and CertLedger that introduce another level of centralization in the blockchain. In Cecoin and X509Cloud blockchain miners are actually CAs. This makes decentralization limited because it allows only entities responsible for issuance of certificates to extend the blockchain and it also increases the possibility of collusion attacks because a smaller number of entities protects the blockchain. Another example of centralization is existence of 'Board Members' in CertLedger which decide which CAs are allowed to issue certificates. This entity seems to be in a similar position as browser and OS vendors in the current CA-based PKI model where they keep lists of trusted CAs.

IoT use PKI based blockchain systems find applications in internet of things. In this area, especially in the case of vehicular networks, there exists a need for privacy to prevent tracking of the vehicles by malicious actors. For this reason, PB-PKI and BARS prevent adversaries from linking public keys to the entities behind them. However, if an entity (e.g. a vehicle) behind some public key starts misbehaving, its identity has to be revealed. For example, a major weakness of PB-PKI is

PKI System	Gas Cost	Ether Cost	USD Cost
EthIKS[1]	367,535	0.0007351 ETH	$0.13011
SCPKI Full	1,044,487	0.0020890 ETH	$0.36975
SCPKI Light	540,185	0.0010804 ETH	$0.19123
IKP	628,640	0.0012573 ETH	$0.22254
Yakubov et al.	2,815,997	0.0056320 ETH	$0.99686
Ghazal	2,402,563	0.0048051 ETH	$0.8505

Table 5.2: Contract deployment costs of different systems. Gas costs are taken from papers or estimated using Remix IDE if we could find an implementation [92]. Ether and USD costs were calculated using [44] on April 9, 2019 using gas price of $2 * 10^{-9}$ ether which was accepted in approximately $\frac{1}{3}$ of the last 200 blocks. It has to be taken into account that a user can offer to pay more gas in order to get the transaction to run earlier.

that any network participant can interfere with another one by revoking her public keys. Therefore, there are schemes through which the identity behind a public key can be revealed. BARS has trusted third parties LEAs (Law Enforcement Authority) that know real identities behind public keys, while PB-PKI requires the majority of users to demand the discovery of the link between identity of malicious entity and her public key for a link to be revealed. PB-PKI's solution offers higher decentralization level at some cost of efficiency. Further, a user in PB-PKI has a higher configuration overhead because it has to take care of the offline keys which if stolen can be used to reveal their identity. Moreover, while PB-PKI offers two privacy levels which affect the trade-off with security, BARS incorporates a reputation system that is in case of disputed misbehavior reports influenced by LEAs' decisions. BARS offers pseudonymity, the same level of privacy as Bitcoin platform. Collusion attack where a vehicle is used to increase the reputation of another vehicle has not been disproved, but included reward and punishment parameters in BARS can be adjusted.

Gas Costs Systems based on Ethereum have to account for gas costs. As contract deployment is usually the costliest operation in smart contracts, here we look at its cost. All the systems in Table 5.2 except IKP and Yakubov et al. are implemented using a single smart contract. IKP's cost in the table refers to the cost of deployment of the central IKP contract that provides all relevant PKI functions, while in the case of Yakubov et al. presented cost refers to deployment of a smart contract that has to be deployed by every CA. BlockPGP is not listed because its contracts are supposed to run on an independent permissioned Ethereum network without gas costs.

Gas costs do not seem to present a problem for adoption of blockchain-based PKI systems. Though differences between the systems seem significant in gas, when converted to USD they are unsubstantial. As miners decide which gas price they are going to offer, we assume that they base it on

[1] Costliest operation 'Create Tree' is used instead of Deployment

the amount of USD that is needed to buy the computational power and pay for the electrical power needed to execute smart contract functions. For this reason, we assume that gas price will with some delay adapt to the changes in Ether / USD exchange rate.

Systems based on CA-based PKI

Most of these systems either extend or modify the operations of web PKI. Only Cecoin is not based on Ethereum, but it therefore uses external structures and logs executed operations to blockchain. The reason for this might be the fact that they decrease trust into CAs through decentralization, automation and quick reactions to activities of parties outside the system. For this they need programmability, and smart contracts offered by the Ethereum platform provide flexibility needed for development of incentive mechanisms, automatic reactions, and automation of certificate verification. Furthermore, all the systems except X509Cloud are based on open permissionless blockchains. This makes them transparent and allows anyone to take part in the system, e.g. in IKP one could play a role of Detector, in Cecoin, CertLedger one could be certificate owner.

Even though BARS, X509Cloud and other systems extend CA-based PKI model, they concentrate on different types of entities. Most systems are oriented more towards web PKI where certificates bind domains with public keys, whereas X509Cloud issues client certificates that bind public keys to user identifiers. BARS is used in vehicular networks and is therefore the only one that hides links between vehicles and entities, while X509Cloud concentrates specifically on the issuance of client certificates. X509Cloud is thus similar to PGP-based systems, but it uses CAs to verify the users. CAs used in X509Cloud are different from CAs used in other systems. In X509Cloud organizations such as universities issue certificates for the users that are associated with them. This way verification burden on CAs is decreased because a CA already knows information that it verifies about the user. Another particular feature can be found in BlockPKI which allows its users to get their certificate signed by multiple CAs. This mitigates unauthorized certificates because attacker that wants to obtain an unauthorized certificate has to compromise several CAs in BlockPKI, and MitM attack during certificate issuance has to be executed against multiple CAs at the same time.

There is a trade-off between automation of the systems and types of certificates offered. On the one hand, BlockPKI and Cecoin automate issuance of certificates and are therefore limited to DV certificates. A certificate in BlockPKI can be issued in two minutes [40]. On the other hand, system by Yakubov et al. and CertLedger allow CAs to issue certificates separately from the blockchain. After issuance CAs only have to add certificates to the blockchain. Third option can be found in IKP where system offers automated reactions to misbehavior, but proof of domain ownership is still done outside the system.

As X.509 certificate format is used most widely on the web, we assume that systems which use it have a better chance of adoption. X.509 format is used in all systems except BlockPKI and BARS. Even X509Cloud's client certificates are stored in X.509 certificate format. Furthermore, similarly to CT, system by Yakubov et al. uses extension fields of the certificate to include additional information needed for blockchain into the certificate. This allows inclusion of such certificates in the current web PKI without any modifications on the web server side. Conversely, BlockPKI uses a custom certificate that points to transaction that contains certificate data and includes proof of certificate inclusion into blockchain.

Systems also differ in their flexibility and compatibility with current CA-based PKI extensions. For example, IKP is compatible with public key pinning and CT, and it can even leverage them by introducing needed checks in check contracts. CT could also be combined with the proposal of Yakubov et al. Their hybrid X.509 certificate could be further extended using X.509v3 extension fields that could include data related to CT.

Compared to blockchain based PKI solutions that improve CA-based PKI, public key pinning extension seems to be extremely rigid. It reduces MitM attack surface, but increases the consequences of attack if adversary intercepted first connection establishment. Blockchain-based systems do not suffer from such problem. Legitimate public key changes cause websites to break when public key pinning is used because the key is pinned on the client-side, but this is not the case in DANE, CT, and blockchain-based PKIs. Conversely, DANE seems to have more similarities with authoritative blockchain-based PKIs because it allows user to pin public key or a certificate to a domain on domain registration.

CT does not prevent unauthorized certificates, it only allows users to detect them. Further, CT is centralized because there is a single entity which maintains a list of authorized log operators. Blockchain-based systems that use CAs however only maintain one log on blockchain whose state is promulgated among the network participants through a gossip protocol. CT is also vulnerable to split-world attacks where Alice is shown a different view of the log compared to the rest of the users. Split-world attacks in CT are possible if there is a collusion between CAs and log operators, but in blockchain-based systems this can only happen through eclipse attack on the blockchain. Moreover, existence of multiple logs in CT also strains entities that have to monitor them. Another feature that is missing in CT compared to blockchain-based systems is reactions to discovered misbehavior. Both IKP and CertLedger detector allow entities to report CA misbehavior which if found genuine results in the revocation of a certificate. Finally, CT does not handle revocation, whereas all blockchain-based systems except BlockPKI usually store revocation status on blockchain.

PGP-based Systems

These systems usually implement PGP key server on the blockchain because this was the only part of PGP infrastructure that was centralized. Using blockchain as a key server makes PGP completely decentralized because it not only uses WoT architecture, but distribution of keys is done in decentralized fashion. Furthermore, in conventional PGP synchronization of PGP key servers is inefficient and could take up to 30 h [124]. This means that what could happen is that revoked certificates are still used even after revocation. Building a PGP key server on blockchain significantly decreases danger of that vulnerability because synchronization then depends only on time between two blocks. Depending on the blockchain this could be as low as 15 secs in Ethereum. In the case of the system by Wilson and Ateniese based on Bitcoin this is approximately 10 minutes which still presents a significant improvement over traditional PGP.

Introduction of blockchain to PGP does not affect problems mentioned in section on centralization. Blockchain-based systems can only try to increase the number of documents, online accounts, etc., which they associate with a public key. The amount of trust in any specific organization can be decreased that way. SCPKI system does exactly this by allowing users to upload pieces of information and letting others sign them. This allows higher granularity of trust as users can

sign only those attributes associated with Alice which are known to them. Finally, usability is another major problem of PGP (Section 2.3 on page 28) that is not influenced by the introduction of blockchain.

Other blockchain-based PGP systems allow only signing of certificates. However, BlockPGP and system by Wilson and Ateniese use standardized OpenPGP certificates whose use might prompt faster adoption rates. BlockPGP adds Ethereum account address in the comment field of OpenPGP certificate. This way they associate the certificate with the Ethereum account. Wilson and Ateniese make a system robuster by including two Bitcoin addresses into the certificate, one for identity verification and another for revocation. As addresses do not have to be the same, if someone compromises the entity in a way that it cannot access its Bitcoin address, distinct revocation address still makes revocation of the certificate possible. Nevertheless, SCPKI's general nature also allows users to include OpenPGP certificates as attributes and additionally authors mention the possibility of adding proofs through which users can link their PGP keys with Ethereum accounts.

Wilson and Ateniese address the problem of quality of PGP signatures, e.g., users might start behaving irresponsibly and sign certificates without actually checking the identity behind the public key. They address the problem by ensuring the signer follows established procedures through identity-verification transactions. Similarly, Authcoin allows only 'bidirectional' signatures where two users sign each other's public keys. Other systems also expect that certain procedures are followed, but they do not give incentives to signers to sign other certificates. BlockPGP and Authcoin do not have incentive mechanism, and SCPKI only has gas costs incentivize miners.

BlockPGP and Authcoin are based on independent blockchain networks. Particularity of BlockPGP is also that the underlying Ethereum is modified in such way that it uses Proof of Authority consensus algorithm instead of PoW. This way there are no computational costs, but the system is centralized in the hands of administrator. Authors also had to change the consensus mechanism as the independent network would not have enough hash power to protect against 51% attacks.

All systems allow users to revoke their keys and signatures they made.

CONIKS-based Systems

Introduction of blockchain to CONIKS renders a separate gossip network between auditors and clients redundant. Length of the intervals between publications of STRs presents a trade-off between security and scalability. Shorter intervals result in faster update times but also increase monitoring cost. To allow higher interval times identity providers in CONIKS issue temporary bindings that act as a promise that the user's key will be included in the next STR [79]. However, STR update times in EthIKS, Catena, and Conifer depend on the block time of the underlying blockchain. As EthIKS is based on Ethereum, trees can be updated every 15 seconds, thus, identity providers do not have to issue temporary bindings. Conversely, updates in Catena and Conifer are much slower because in Bitcoin time between blocks is 10 minutes. EthIKS' use of smart contracts on Ethereum means that identity provider's calls to update function are logged on the blockchain. This might leak information.

In contrast to Catena, which implements CONIKS on top of Catena logs by only storing STRs to blockchain, Conifer changes the way key directory's state is represented in STRs. While in CONIKS and Catena STRs represent signed roots of the current state of key directories, in Conifer each

operational tree from which STR is created contains changes applied in the time interval between two STRs. This enables users to audit during lookup and reduces the need for monitoring. EthIKS is similar to Conifer in this regard because a key directory is stored in a smart contract whose state is included in every block in a way that it either points to the previous block or includes changes that result from application of transactions in a block [20]. Furthermore, compared to EthIKS, Catena expects users to take part in HRN network as well. This allows users to run lightweight clients without straining the Bitcoin network. Nevertheless, similarly to conventional CONIKS, blockchain-based systems rely on existing PKI for public keys associated with identity providers.

Authoritative Systems

Authoritative systems fulfill all three properties of Zooko's Triangle (Appendix A.4 on page 95). They offer human-meaningful names and blockchain provides decentralization. Security property is fulfilled through blockchain's append-only feature as well. Comparable extension of CA-based PKI is DANE, however it only provides human-meaningful names and security. Although DANE offers the possibility to circumvent CAs, it does not fulfill distributed property because zones in the DNS tree are operated by trusted third parties, and there is a single root of trust.

Most of the authoritative systems operate similarly. They offer users to store name-value pairs where public key is stored as a value. All of the systems implement functionality to add, update, look-up, and revoke key bindings, but as Certcoin and PB-PKI link updated public key to a previous one (Section 4.5.1 on page 56) and use the notion of online and offline keys, this allows users of such systems to recover lost private keys. Particular feature of PB-PKI is unlinkability between public keys and identities behind them, while Nidaba proposes a way to address the scalability problem of the underlying blockchain (Chapter 5 on page 62).

In general, authoritative systems do not associate names with certificates but raw public keys. Like other naming systems, authoritative systems are susceptible to domain squatting. Namecoin tries to prevent it with registration fees, but Kalodner et al. found in 2015 only 28 domains considered to be unaffected by squatting [63]. However, Namecoin developers claim that websites using both DNS and Namecoin should not be considered squatted which would mean that there are additional 111 domains that are not squatted [84]. Considering that the initial study covered 120,000 names, this still means that only somewhat more than 1‰of domains are not squatted. While Ghazal also uses registration fees to prevent squatting, developers of Blockstack offer an alternative solution which includes a pricing function that makes the price of names dependent on certain properties of the name such as inclusion of numeric characters.

6 TKI

In this chapter Transparent (Public) Key Infrastructure (TKI) is presented. TKI is a system that was developed in the course of this thesis. It extends the current CA-based PKI system by allowing users to post key bindings on blockchain and record which key bindings they encountered on a domain. By allowing users to establish trust relationships like in WoT architecture, sybil attacks on TKI are mitigated. Prototype of TKI is built on top of Ethereum platform and makes use of a single smart contract. Extensive use of programmability was inspired by IKP [77] and Ghazal [80]. The underlying concept is based on the notion that the public key most often encountered whilst communicating with some entity, in fact, presumably belongs to that entity. Browser extension included in TKI shows users which public keys were encountered by other clients so that they can detect some subset of MitM attacks.

TKI is built on Ethereum because it is the largest blockchain platform that also offers the possibility to execute smart contracts which allows development of more complex systems. There is a large number of miners and compared to numerous other blockchains it is actively developed and maintained.

As TKI allows detection of anomalous key bindings on websites, it can be considered to belong to the category of blockchain-based PKI systems that extend CA-based PKIs, but the possibility of TKI users to enter in trust relationships with each other also associates it with PGP-based systems. Part of TKI that tries to decrease the trust in CAs by logging key bindings is inspired by CT [23]. On the contrary, the use of WoT architecture and possibility of associating arbitrary information including PGP keys with Ethereum accounts is inspired by SCPKI's use of WoT to allow users to associate pieces of information with their Ethereum addresses and to sign data associated with other accounts [9].

Prototype consists of three parts:

- **Smart Contract** Blockchain contract is the main part of the system. It defines functions that users can employ to interact with the blockchain and store information on it. Smart contract is written in Solidity, high-level programming language compiled in EVM code, low-level bytecode which runs on Ethereum Virtual Machine (EVM) [20].
- **Client Application** Command line application through which users can interact with the smart contract. They can verify key bindings and access data stored on blockchain.
- **Browser Extension** Firefox extension that allows users to post novel key bindings to blockchain as well as to log those key bindings that they encountered while visiting a domain. Everyone with an installed browser extension can see information related to all certificates encountered on a website. Browser extension is limited to Firefox because it is the most popular browser that provides an API for browser extensions to access the certificates during connection establishment.

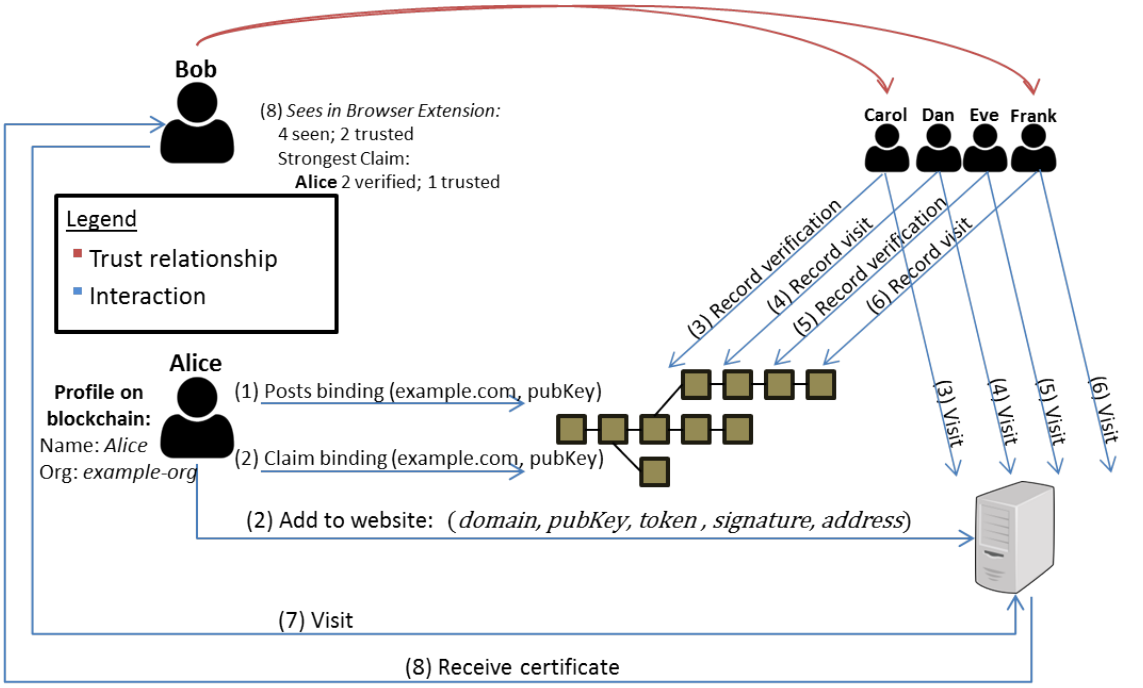

Figure 6.1: Use of TKI on the web. In the figure we assume that the public key is the same on all three occasions: (1) in the certificate that Bob receives; (2) seen and/or verified by Carol, Dan, Eve and Frank; (3) public key contained in the key binding which Alice posted and claimed. Signature posted to a website is created using a private key $\hat{k}_{Eth}$ corresponding to the Ethereum *address*: $signature = T((domain, pubKey, token), \hat{k}_{Eth})$.

One way in which users could interact in TKI can be seen in Figure 6.1. Every Ethereum account that registers in TKI has an associated profile that is stored on blockchain and contains identifying information, e.g. Alice has associated name and organization information with her account. If Alice owns a website *example.com*, she might want to store her key binding in TKI and immediately claim it. In order to allow other users to verify Alice's claim and through it associate her Ethereum account with the key binding, Alice signs $(example.com, h(k), token)$ tuple, where $h(k)$ is the hash of public key, using the private key of her Ethereum account and posts the tuple with the signature to blockchain.

Since the user that posts a key binding to blockchain has to pay for gas costs, the reward for posting should be higher than gas costs in order to incentivize users. TKI contract maintains a special reward fund for this purpose. Every user that can benefit from TKI has an incentive to send some ether to TKI contract because ether collected this way is used to reward users that post key bindings.

Registered TKI users can record their 'visits' and 'verifications'. Visit refers to an act of encountering a specific key binding on a domain in certificate, whereas during verification user checks whether a signature created with an Ethereum account of one of the TKI users that claim to own the key binding exists on a specific path on website. Visits are recorded for a key binding only if user has chosen to record them, and the key binding is already posted to blockchain. If a key binding is not posted to blockchain, the user can choose whether to post it or not. When a user posts a key binding, she gets rewarded from funds designated for that use. Poster of a key binding has to define

a ‘reward’ structure that is linked to her and the key binding. In the reward definition she has to include ether amounts for visits and choose how much ether to send for initial balance of the reward. To safeguard against spammers that could post nonexistent key bindings solely with the goal of obtaining posting rewards, TKI includes a minimum initial balance that has to be defined for the reward created when posting a key binding.

As TKI is built on a permissionless blockchain, a single user can have many accounts which she can register to TKI system. For this reason, TKI users can enter into trust relationships with each other which allows them to recognize sybil attacks that record fake visits and verifications. In Figure 6.1, when Bob visits *example.com* he can see in his browser extension that four TKI users have seen the same key binding which he sees when they visited *example.com*. Since some of these users may be created by a sybil attacker, TKI also shows Bob that two users whom Bob trusts have also seen the same key binding. Similarly, in the case of verifications Bob sees that two TKI users, of which one is trusted, have verified Alice’s claim on the key binding.

6.1 Functionality

In this section the functionalities offered by TKI are described. Users can register, update their information, post key bindings, create rewards, and associate them with key bindings. They can also claim that they own a key binding as well as send ether to the contract to show how much they trust someone.

User Registration User can register in TKI system by associating her Ethereum account with a chosen name and arbitrary data which represent user’s profile in TKI and can be updated at any time. In order to distinguish registered from unregistered entities, registered entities will be called ‘TKI users’. Although TKI prototype does not do any checks on registration, having a separate registration function allows the system to introduce mechanisms to prevent sybil attacks. For instance, one could include a procedure where a certain number of existing TKI users have to decide on whether to accept someone’s registration. Use of such measures could also increase trustworthiness of visits and verifications of all TKI users and reduce the need to rely on existing trust relationships which would especially affect users of websites that have few visitors.

Users that do not register can also utilize information collected by TKI through a browser extension. Besides, they could see information about the relationships between users through command line application. However, as data on visits and verifications collected that way is insufficiently trustworthy if TKI registration does not contain above-mentioned mechanisms, TKI users can also see number of visits and verifications recorded by their trusted users. Unregistered users cannot benefit from this feature. In the prototype users can register through browser extension or command line application.

There is a trade-off between privacy and trustfulness of TKI users. TKI users that have more identifying information may be more trusted, but a certain level of privacy can be preserved by linking only other internet accounts to a TKI account. If users for example run a popular blog, they could include in their user information on TKI a link to a blog post where they state their ownership of the Ethereum account.

Posting Key Bindings All TKI users can post any key binding they encounter to blockchain. Poster of the binding is recorded, and she has the possibility to define a reward structure. In order to decrease gas costs, key bindings on blockchain are stored with a hash of the public key. Ether sent with transaction is added to the created reward's balance. Furthermore, there is also a possibility to associate a certificate hash with the key binding. In the case of a domain key binding, if the user is the first one to encounter a specific key binding, browser extension automatically stores seen X.509 certificate on IPFS, while certificate hash is stored on blockchain. TKI could also be used with other types of key bindings. It could, for example, store client or OpenPGP certificates. In that case, however, visits and verifications would have another interpretation.

Trust Relationships TKI users can indicate that they trust another TKI user by sending a certain amount of ether to TKI contract. This is similar to the system by Wilson and Ateniese, although in the case of TKI sent ether is not recoverable [120]. There is also the notion of trust levels familiar from PGP (Section 2.3.1 on page 28). For instance, Alice can choose a trust level that describes how much she trusts Bob to honestly record visits and verifications. The amount of ether sent in the name of a trust relationship is distinct from the chosen trust level. Similarly to SCPKI, as users can associate any arbitrary information with their TKI accounts to increase their trustfulness, they could also associate their PGP keys or e-Citizen identities by adding a public key and a signature of the Ethereum address created with a private key associated with the other identity to their TKI user profile. In a similar way, users could create recovery TKI accounts and associate them with their main TKI account so that if the private key of the main account gets stolen, all of the trust relationships can be transferred to the recovery account. While Sybil attacks can be mitigated with established trust relationships, they cannot help against MitM attack because trusted users can also be a victim of MitM attack.

Claims and Revocation TKI user can claim that she owns a certain key binding (Figure 6.1). In order to support her claim, she should sign a domain, public key, and token chosen during the creation of claim using a private key that corresponds to her Ethereum address. This information should be posted on a specific path on the website to allow other TKI users to verify the claim. The number of times a claim is verified represents the belief in the link between a key binding and Ethereum account. For this reason, claimant with the most recorded verifications can declare that the key binding is revoked. If after the loss of private key a malicious CA does not want to revoke the certificate associated with the key binding, domain owner can this way indicate that the key binding is revoked provided that she has the most verifications. Nevertheless, since verifications are insufficiently trustworthy if there is no registration procedure, other TKI users can again decide whether to trust the revocation declaration because they can see verifications of their trusted users.

Creating and Updating Rewards A reward structure in TKI contains information about the reward issuer, amount of funds paid into a reward, and values that define how much ether is paid out when a visit or verification is recorded. Although rewards associated with a claimant or poster are created automatically when a claim is made or key binding is posted, any other TKI user can create a reward and associate it with a key binding. Once created rewards can be updated by sending additional ether for the reward fund and by changing the amount of ether paid out on visits and verifications.

Visits and Verifications TKI user can choose to record visits and verifications on blockchain. In TKI prototype every user can record only one visit and verification for a specific key binding, but it might also be appropriate to only prevent users from recording visits for a certain time period after they record a visit. To record a visit means to simply note in blockchain that a certain TKI user has encountered a certain key binding. Conversely, during verification a user verifies the signature found on the website that is signed with a private key of some Ethereum account belonging to a TKI user that claimed the key binding and increases the counter which represents the number of times a corresponding claim has been verified. TKI users who record visits and verifications should be rewarded not only because other users benefit from the information they share, but also because sharing this data diminishes their privacy.

6.2 Security

Here we informally analyze TKI with respect to Sybil attacks and MitM attacks of different strength. The assumptions that we make in the course of this section are that the honest miners hold more than 50% of computational power protecting the blockchain and that the used cryptographic primitives are secure. In the case of Ethereum blockchain that underlines TKI, at the time of writing no single entity holds more than 50% of the hash power. We also assume that the adversary is able to obtain an unauthorized certificate from CA.

With regard to the security of public keys associated with the user's profiles, if the user's private key of their posted public PGP key gets stolen for example, she can update the relevant information in her TKI profile. On the other hand, if she does not trust another TKI user anymore, she can also decrease the level of trust that she associated with that user.

6.2.1 Sybil Attack

In a sybil attack an adversary creates many Ethereum accounts and registers them in TKI. We assume that the attacker wants to deceive Alice so that she believes an unauthorized certificate is safe. We also assume that the certificate presented to Alice is not revoked, and thus the browser's UI does not show any warning signs to Alice.

Attacker uses registered fictional accounts to record many visits and verifications for a key binding found in the unauthorized certificate. When Alice receives such a certificate, she finds that the key binding that she encountered is the most often seen key binding. Furthermore, she sees a claim for this binding which is verified many times. Since in this situation Alice would believe that the received certificate is valid, TKI allows Alice to define a set of TKI users to whom she trusts. Consequently, browser extension also includes the number of trusted users that have seen this key binding and verified the claim with the most verifications. If Alice carefully chooses trusted users, she will notice a discrepancy between the experience of trusted users and the other visitors which could point to the fact that fictional users were used to increase the number of visits and verifications associated with the encountered certificate.

6.2.2 MitM Attacks

We analyze the effects of MitM attacks on the system and mitigation provided by TKI depending on the position and strength of the adversary in the network. On the one hand, if an MitM attacker is located near a specific client, the damage she can make is limited to that client. We call this a 'weak' MitM attack. Attacker can become MitM near Alice through ARP or MAC address spoofing in her data link layer network. On the other hand, an attacker considered to be in front of the server can be someone who runs ARP or MAC address spoofing on the server in its data link network, or it can be a stronger attacker that controls an autonomous system and can intercept most of the messages intended for the server through BGP path forgery. Attacker that can influence DNS servers can also be considered to be in front of the server. We call such attack a 'strong' MitM attack. Next, we look at three scenarios depending on whether the domain owner uses TKI.

Steady Case Scenario In a 'steady case scenario' a key binding is posted to the blockchain and an actual domain owner has claimed the binding. After some time most visitors of this website encountered the public key claimed by the domain owner whose claim also has the most verifications. If a TKI user is now subject to a weak MitM attack, she can notice immediately that the key binding which she encounters does not have a lot of visits. Conversely, if a strong MitM attacker starts attacking the domain, the number of visits to attacker's key binding starts rising, and if the adversary claims the key binding, verifications start growing as well. This indicates an anomaly and visitors should check TKI profile of the strongest claimant, i.e. the domain owner in this case. To mitigate the attack domain owner can add information to her TKI profile that says the domain is under attack. Other TKI users can inspect TKI profile of the attacker as well.

Just Registered Scenario We assume that some key binding for the domain in question has already been posted to blockchain. If a domain owner has just registered to TKI, she can immediately claim that she owns a certain key binding associated with the domain. Weak MitM attacker still has a limited influence on the system because he cannot collect a significant number of visits. Such an attacker may be able to trick users if the domain is not popular. In contrast, a strong MitM attacker that has claimed the key binding might have already collected many visits and verifications and in the course of the attack might have deceived many users. To mitigate the attack, actual domain owner can add to her TKI profile information about the situation and as much as possible evidence that she is the owner of the domain. For example, if the attacked website belongs to a company, corresponding TKI profile might include other ways in which they can be contacted.

Not Registered Scenario There is a possibility that some user posted Alice's key binding to blockchain, so even if Alice is not a TKI user, her visitors can benefit from the system. TKI also gives an ability to the poster, even if he is not a domain owner, to offer rewards. This allows various other parties, apart from the domain owner, who have interest in the security of a domain to offer rewards. However, there is also the possibility that Eve, a malicious TKI user will post and claim some key binding associated with Alice's domain. If Eve mounts a successful MitM attack on many of the connections to Alice's web server, Eve can gain many verifications and visits to her key binding. This may make Eve's claim over a domain seem trustworthy, and depending on the number of times Alice's key binding was seen, it may even seem to clients registered with TKI that Alice's key binding belongs to the attacker.

Operation	Approximate Gas Cost		
	Gas	**Ether**	**USD**
register	146, 231	0.0002925 ETH	$0.04739
updateUserInfo	47, 688	0.0000954 ETH	$0.01545
addToPostFund	43, 195	0.0000864 ETH	$0.014
postKeyBinding	289, 302	0.0005786 ETH	$0.09373
createBindingReward	133, 410	0.0002668 ETH	$0.04322
updateBindingReward	46, 702	0.0000934 ETH	$0.01513
claimKeyBinding	189, 493	0.0003790 ETH	$0.0614
updateClaim	52, 673	0.0001053 ETH	$0.01706
visitedKeyBinding	133, 641	0.0002673 ETH	$0.0433
verifiedClaim	125, 531	0.0002511 ETH	$0.04068
revokeKeyBinding	45, 615	0.0000912 ETH	$0.01477
initTrust	168, 764	0.0003375 ETH	$0.05468
updateTrust	37, 873	0.0000757 ETH	$0.01226
Contract Deployment	4, 721, 415	0.0094428 ETH	$1.52973

Table 6.1: Gas costs of functions offered by TKI smart contract approximated using Remix IDE [92]. Ether and USD costs were calculated using [44] on May 1, 2019 using gas price of $2 * 10^{-9}$ ether which was accepted in approximately 59% of the last 200 blocks.

6.3 Evaluation

Prototype of TKI was developed using Truffle Suite, a popular development environment used in the development of distributed applications that run on Ethereum platform [113]. During development TKI smart contract was deployed on Ganache, a private blockchain that is part of the Truffle Suite.

Gas costs associated with functions offered by the smart contract are presented in Table 6.1. Costs of all operations except contract deployment are under $0.1 which seems reasonable compared to costs of certificates in CA-based PKI. The highest cost is incurred on contract deployment which is executed only once. Consequently, one may set a lower gas price on contract deployment as one could afford waiting longer until the corresponding transaction is processed when starting the system. Gas costs of other operations could perhaps be lowered further by storing less information in contract storage and relying instead on clients to filter logs which are stored on blockchain when events in operations are fired.

Gas costs associated with *register* and *updateUserInfo*, operations that change the profile of a TKI user, depend largely on the amount of data sent to blockchain with them. In Table 6.1 we sent three pieces of information with these functions to approximate their gas costs. Further, the weakness

	Stored with IPFS	Stored on blockchain
Gas cost	289, 302	2, 847, 995
Time to post key binding		
Best	664 ms	385 ms
Mean	1, 118 ms	431 ms
Worst	2, 244 ms	531 ms

Table 6.2: Timing results were obtained by storing the certificate found on *example.com* 10 times on blockchain and IPFS.

```
lapi@lapi:~/TKI/tki/contracts$ myth -x Tki.sol
The analysis was completed successfully. No issues were detected.

lapi@lapi:~/TKI/tki/contracts$
```

Figure 6.2: Results of security analysis with Mythril Classic

of having separate independent rewards is that all of them have to be considered when a visit or verification is recorded. As we loop through stored rewards, there is an upper limit on the number of active rewards associated with a specific key binding at the same time because of the possibility of exceeding the block gas limit.

Since high gas costs associated with posting a key binding would also require higher rewards for posters, we store only the hash of the certificate associated with the posted key binding to blockchain. Certificate hash links to the certificate stored on IPFS (Appendix A.5 on page 95). We find that compared to storing the certificate on blockchain, using IPFS reduces gas costs to only a tenth of the otherwise needed gas costs (Table 6.2). In contrast, time needed to post a key binding and the certificate is on average more than twice as long, and variance in the needed time is considerably higher in the case of IPFS.

```
INFO:symExec:   ====== Analysis Completed ======
INFO:root:contract Tki.sol:Tki:
INFO:symExec:   ============ Results ===========
INFO:symExec:     EVM Code Coverage:                  32.2%
INFO:symExec:     Integer Underflow:                  True
INFO:symExec:     Integer Overflow:                   True
INFO:symExec:     Parity Multisig Bug 2:              False
INFO:symExec:     Callstack Depth Attack Vulnerability:  False
INFO:symExec:     Transaction-Ordering Dependence (TOD): False
INFO:symExec:     Timestamp Dependency:               False
INFO:symExec:     Re-Entrancy Vulnerability:          False
```

Figure 6.3: Results of security analysis with Oyente

As bugs in smart contracts can have wide reaching consequences (e.g. DAO contract on Ethereum [47]), we have analyzed prototype TKI contract using the security analysis tools Mythril Classic developed by ConsenSys [82] and Oyente proposed by Luu et al. [74]. Through use of symbolic execution and static code analysis these tools help developers discover vulnerabilities and potential security bugs in their contracts. Generic analysis done with Mythril Classic found no vulnerabilities in the contract (Figure 6.2), whereas Oyente found integer overflow and underflow vulnerabilities (Figure 6.3), albeit all arithmetic operations in the contract are done using functions included in SafeMath library from OpenZeppelin that reverts transaction if arithmetic operation results in an overflow [98].

7 Conclusion

To conclude, we look at the advantages and disadvantages of the four groups of PKI systems based on blockchain.

- **CA-based PKI with Blockchain** Main advantage of these systems seems to be easier adoption as they only extend the currently most widely used PKI. Single point of failure problem associated with CAs is mitigated because these systems make CAs accountable as they log information on blockchain. The use of blockchains also offers a solution for revocation because revocation status can be stored on blockchain so that vulnerable CRL and OCSP methods do not have to be used. Furthermore, there are systems that offer automatic reactions to misbehavior which decreases the time during which vulnerability to impersonation attacks exist. Incentive mechanisms could also result in the higher number of entities monitoring CAs. On the other hand, CAs still have significant influence on PKI and except revocation problem, other problems, although mitigated are still present (Section 2.2.2 on page 26). Finally, only monitoring of the PKI is decentralized.
- **PGP with Blockchain** Blockchain allows decentralization of key servers, centralized part of PGP. This makes these systems completely decentralized. Although transition to these systems might be harder compared to CA-based PKIs, user community that uses PGP might be more open to changes and the use of blockchain technology. Nevertheless, the use of blockchains does not affect interactions between users who might accidentally sign public keys of malicious entities. Usability problem of PGP also remains.
- **CONIKS with Blockchain** Blockchain network takes over the place of gossip network between auditors and monitors which are also rewarded through transaction fees. As CONIKS stores snapshots of key directories in form of their root hashes, scalability problem of blockchains is not that pronounced, but identity provider's control of key directory makes these systems centralized.
- **Authoritative Systems** These systems seem to offer the highest level of security because they are completely decentralized and minimize indirection. Bandwidth between the client and the server on connection establishment can be reduced and there is no need for monitoring entities. Regardless, the main weakness of authoritative systems is that they are restricted to specific namespaces that are not supported by DNS, and they do not protect current domains.

Blockchain seems to be suitable for implementation of PKI, but it also seems that it is still too early to make an assumption that blockchain is decentralized and secure (Section 3.3 on page 41). While increase in computational power protecting PoW blockchains might bring higher security it comes at high environmental costs. Ethereum is, however, an example of a large blockchain that aims to transition to PoS. Largest permissionless blockchains also have scalability problems. Although this could be circumvented through specific designs of blockchain-based PKI like the use of separate constructions and inclusion of several operations in a single transaction, scalability

increase of blockchains themselves is an active area of research [19]. As seen in the example of MultiChain, permissioned blockchains support higher number of transactions per second and allow the use of more environmentally friendly consensus procedures. Trade-off between centralization and scalability implies the necessity for reconsideration of the need for decentralization.

TKI, a PKI system developed in the course of this thesis combines CA-based PKI with WoT on blockchain. To make it easier for adoption and more practical, TKI contains a browser extension which allows clients to post key bindings to blockchain and associated certificates to IPFS as well as log encountered key bindings to provide detection of anomalous key bindings and mitigate the risk of MitM attack. Registered users can associate information with Ethereum accounts and enter into trust relationships with others which helps prevent sybil attacks.

In this thesis the introduction of PKIs was motivated through a key binding problem. PKIs most often found in practice as well as PKIs that underlie blockchain-based PKI systems were then introduced. Likewise, extensions of CA-based PKIs were looked at. Thereafter we looked into blockchain, its structure and functionality, different types of blockchains, attacks on it, and the way PKI is established on it. Afterwards, PKI systems based on blockchains were classified, four main categories were determined, and two systems from each category presented. Discussion that followed touched on security, adoption, underlying blockchains, and centralization. Finally, TKI was presented and its functionality and security were informally analyzed.

Bibliography

[1] A. Abdul-Rahman. "The pgp trust model". In: *EDI-Forum: the Journal of Electronic Commerce*. Vol. 10. 3. 1997, pp. 27–31. URL: https://pdfs.semanticscholar.org/e9aa/5d8032c1d925ea6a02dd3be93f42e831c965.pdf (cit. on p. 29).

[2] *About Buypass*. URL: https://www.buypass.com/ssl/about-us (visited on 04/19/2019) (cit. on p. 23).

[3] M. Ali, J. Nelson, R. Shea, M. J. Freedman. "Blockstack: A Global Naming and Storage System Secured by Blockchains". In: *USENIX Annual Technical Conference*. USENIX Association, 2016, pp. 181–194. ISBN: 978-1-931971-30-0. URL: https://www.usenix.org/conference/atc16/technical-sessions/presentation/ali (cit. on pp. 43, 46, 55, 57, 60, 63, 64).

[4] L. Axon, M. Goldsmith. "PB-PKI: A Privacy-aware Blockchain-based PKI". In: *Proceedings of the 14th International Joint Conference on e-Business and Telecommunications*. SCITEPRESS - Science and Technology Publications, 2017. DOI: 10.5220/0006419203110318 (cit. on p. 46).

[5] L. Backstrom, P. Boldi, M. Rosa, J. Ugander, S. Vigna. "Four degrees of separation". In: *Proceedings of the 3rd Annual ACM Web Science Conference on - WebSci '12*. ACM Press, 2012. DOI: 10.1145/2380718.2380723 (cit. on p. 64).

[6] A. Baliga. "Understanding blockchain consensus models". In: *Persistent*. 2017. URL: https://pdfs.semanticscholar.org/da8a/37b10bc1521a4d3de925d7ebc44bb606d740.pdf (cit. on p. 40).

[7] R. Barnes, J. Hoffman-Andrews, D. McCarney, J. Kasten. *Automatic Certificate Management Environment (ACME)*. RFC 8555. Mar. 2019. DOI: 10.17487/RFC8555 (cit. on pp. 28, 60).

[8] D. Basin, C. Cremers, T. H.-J. Kim, A. Perrig, R. Sasse, P. Szalachowski. "ARPKI: Attack resilient public-key infrastructure". In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM. 2014, pp. 382–393. DOI: 10.1145/2660267.2660298 (cit. on pp. 18, 31).

[9] M. Al-Bassam. "SCPKI". In: *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts - BCC '17*. ACM Press, 2017. DOI: 10.1145/3055518.3055530 (cit. on pp. 46, 52, 53, 71).

[10] J. Benet. *IPFS-content addressed, versioned, P2P file system*. 2014. arXiv: 1407.3561 [cs.DC] (cit. on p. 95).

[11] *Bitmain Corporate Introduction*. URL: https://www.bitmain.com/about (visited on 04/17/2019) (cit. on p. 43).

[12] *Blockchain Statistics*. URL: https://explorer.emercoin.com/stats (visited on 04/07/2019) (cit. on p. 62).

[13] *Blockchain's Once-Feared 51% Attack Is Now Becoming Regular*. URL: https://www.coindesk.com/blockchains-feared-51-attack-now-becoming-regular (visited on 04/13/2019) (cit. on p. 41).

[14] S. Boeyen, S. Santesson, T. Polk, R. Housley, S. Farrell, D. Cooper. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. RFC 5280. May 2008. DOI: 10.17487/RFC5280 (cit. on pp. 17, 23, 24).

[15] J. Bonneau. "EthIKS: Using Ethereum to audit a CONIKS key transparency log". In: *International Conference on Financial Cryptography and Data Security*. Springer. 2016, pp. 95–105. DOI: 10.1007/978-3-662-53357-4_7 (cit. on p. 46).

[16] J. Bonneau. "Hostile blockchain takeovers (short paper)". In: *International Conference on Financial Cryptography and Data Security*. Springer. 2018, pp. 92–100. DOI: 10.1007/978-3-662-58820-8_7 (cit. on p. 42).

[17] BTC.com. *Mining Insights*. URL: https://eth.btc.com/miningstats (visited on 04/17/2019) (cit. on pp. 42, 43).

[18] BTC.com. *Pool Distribution*. URL: https://btc.com/stats/pool?pool_mode=week (visited on 04/17/2019) (cit. on p. 43).

[19] V. Buterin. *Ethereum scalability research and development subsidy programs*. URL: https://blog.ethereum.org/2018/01/02/ethereum-scalability-research-development-subsidy-programs/ (visited on 04/25/2019) (cit. on p. 82).

[20] V. Buterin et al. "A next-generation smart contract and decentralized application platform". In: *Ethereum White Paper* (2019). URL: https://github.com/ethereum/wiki/wiki/White-Paper (cit. on pp. 35, 37, 39, 69, 71).

[21] *CA:Symantec Issues*. URL: https://wiki.mozilla.org/CA:Symantec_Issues#Issue_D:_Test_Certificate_Misissuance_.28April_2009_-_September_2015.29 (visited on 03/11/2019) (cit. on p. 18).

[22] *CA/Included Certificates*. URL: https://wiki.mozilla.org/CA/Included_Certificates (visited on 03/26/2019) (cit. on p. 27).

[23] *Certificate Transparency*. 2011. URL: https://www.certificate-transparency.org (visited on 04/18/2019) (cit. on pp. 32, 71).

[24] J. Chandler. *DigiCert Acquires Verizon Enterprise SSL Business*. 2015. URL: https://www.digicert.com/news/2015-06-23-digicert-acquires-verizon-business/ (visited on 01/08/2019) (cit. on p. 17).

[25] *Chrome Policies*. Accessed: 2019-03-11. URL: https://chromium.googlesource.com/chromium/src/+/master/net/docs/certificate-transparency.md#Chrome-Policies (cit. on pp. 18, 32).

[26] *Chromium Certificate Transparency Log List*. 2019. URL: https://cs.chromium.org/chromium/src/components/certificate_transparency/data/log_list.json (visited on 02/12/2019) (cit. on p. 32).

[27] *Comodo Certification Practice Statement*. 2018. URL: https://sectigo.com/uploads/files/Comodo-CA-CPS-4-2.pdf (visited on 03/11/2019) (cit. on p. 25).

[28] *Confirmation*. URL: https://en.bitcoin.it/wiki/Confirmation (visited on 04/19/2019) (cit. on p. 41).

[29] *Cryptocurrency statistics*. URL: https://bitinfocharts.com/ (visited on 04/06/2019) (cit. on pp. 38, 62).

[30] *Deep Chain Reorganization Detected on Ethereum Classic (ETC)*. URL: https://blog.coinbase.com/ethereum-classic-etc-is-currently-being-51-attacked-33be13ce32de (visited on 04/13/2019) (cit. on p. 41).

[31] *Deprecations and removals in Chrome 67*. 2018. URL: https://developers.google.com/web/updates/2018/04/chrome-67-deps-rems#deprecate_http-based_public_key_pinning (visited on 04/18/2019) (cit. on p. 31).

[32] S. Dick, N. Thomas, F. Ball. "Performance analysis of PGP". In: *Proceedings of 22nd UK Performance Engineering Workshop, Bournemouth University*. 2006. URL: https://www.researchgate.net/profile/Nigel_Thomas5/publication/281971013_Performance_analysis_of_PGP/links/56001b1e08aec948c4fa658f/Performance-analysis-of-PGP.pdf (cit. on p. 28).

[33] T. Dierks, E. Rescorla. *The transport layer security (TLS) protocol version 1.2*. Tech. rep. 2008. DOI: 10.17487/rfc5246 (cit. on p. 17).

[34] *DigiCert Completes Acquisition of Symantec's Website Security and Related PKI Solutions*. 2017. URL: https://www.digicert.com/news/digicert-completes-acquisition-of-symantec-ssl/ (visited on 01/08/2019) (cit. on p. 17).

[35] Y. Dong, W. Kim, R. Boutaba. "Conifer: centrally-managed PKI with blockchain-rooted trust". In: *IEEE International Conference on Blockchain (Blockchain)*. 2018. URL: https://pdfs.semanticscholar.org/117f/0a4710543fea0d78a00f2221d52b0a47e7ee.pdf (cit. on pp. 46, 54, 63).

[36] *Double-spending*. URL: https://en.bitcoinwiki.org/wiki/Double-spending (visited on 04/30/2019) (cit. on p. 41).

[37] V. Dukhovni, W. Hardaker. *The DNS-Based Authentication of Named Entities (DANE) Protocol: Updates and Operational Guidance*. RFC 7671. Oct. 2015. DOI: 10.17487/RFC7671 (cit. on p. 33).

[38] Z. Durumeric, J. Kasten, M. Bailey, J. A. Halderman. "Analysis of the HTTPS certificate ecosystem". In: *Proceedings of the 2013 conference on Internet measurement conference*. ACM. 2013, pp. 291–304. DOI: 10.1145/2504730.2504755 (cit. on pp. 17, 24, 26).

[39] Z. Durumeric, F. Li, J. Kasten, J. Amann, J. Beekman, M. Payer, N. Weaver, D. Adrian, V. Paxson, M. Bailey, et al. "The matter of heartbleed". In: *Proceedings of the 2014 conference on internet measurement conference*. ACM. 2014, pp. 475–488. DOI: 10.1145/2663716.2663755 (cit. on p. 24).

[40] L. Dykcik, L. Chuat, P. Szalachowski, A. Perrig. *BlockPKI: An Automated, Resilient, and Transparent Public-Key Infrastructure*. 2018. arXiv: 1809.09544 [cs.CR] (cit. on pp. 46, 66).

[41] T. Eisenbarth, S. Kumar, C. Paar, A. Poschmann, L. Uhsadel. "A survey of lightweight-cryptography implementations". In: *IEEE Design & Test of Computers* 6 (2007), pp. 522–533. DOI: 10.1109/mdt.2007.178 (cit. on p. 21).

[42] C. Ellison, B. Schneier. "Ten risks of PKI: What you're not being told about public key infrastructure". In: *Computer Security Journal* 16.1 (2000), pp. 1–7. URL: https://www8.cs.umu.se/kurser/5DV153/HT14/literature/ellison2000ten.pdf (cit. on pp. 17, 59).

[43] Emercoin International Development Group. "EMCSSL". In: *Emercoin White Paper* (2015). URL: https://emercoin.com/files/uploads/docs/EMCSSL.pdf (cit. on pp. 46, 59, 61).

[44] *ETH Gas Station*. URL: https://ethgasstation.info/calculatorTxV.php (visited on 04/09/2019) (cit. on pp. 65, 77).

[45] C. Evans, C. Palmer, R. Sleevi. *Public Key Pinning Extension for HTTP*. RFC 7469. Apr. 2015. DOI: 10.17487/RFC7469 (cit. on pp. 31, 32).

[46] T. Fadai, S. Schrittwieser, P. Kieseberg, M. Mulazzani. "Trust me, I'm a Root CA! Analyzing SSL Root CAs in Modern Browsers and Operating Systems". In: *Availability, Reliability and Security (ARES), 2015 10th International Conference on*. IEEE. 2015, pp. 174–179. DOI: 10.1109/ares.2015.93 (cit. on p. 17).

[47] S. Falkon. *The Story of the DAO—Its History and Consequences*. 2017. URL: https://medium.com/swlh/the-story-of-the-dao-its-history-and-consequences-71e6a8a551ee (visited on 04/20/2019) (cit. on pp. 60, 79).

[48] H. Finney, L. Donnerhacke, J. Callas, R. L. Thayer, D. Shaw. *OpenPGP Message Format*. RFC 4880. Nov. 2007. DOI: 10.17487/RFC4880 (cit. on pp. 23, 28).

[49] D. Floyd. *Bitmain By the Numbers: An Inside Look at a Bitcoin Mining Empire*. 2018. URL: https://www.coindesk.com/bitmain-by-the-numbers-an-inside-look-at-a-bitcoin-mining-empire (visited on 04/17/2019) (cit. on p. 64).

[50] *Fraudulently Obtained Genuine (FOG) Documents*. 2017. URL: https://www.itwsecurityd ivision.com/Portals/0/documents/ITW%20White%20Paper%20-%20Fraudulently%20Obtain%20Genuine%20FOG%20Documents.pdf (visited on 04/20/2019) (cit. on p. 64).

[51] C. Fromknecht, D. Velicanu, S. Yakoubov. "Certcoin: A namecoin based decentralized authentication system". In: *Massachusetts Inst. Technol., Cambridge, MA, USA, Tech. Rep* 6 (2014). URL: https://pdfs.semanticscholar.org/7288/9440eaeb7a1a17a8be830feff236b5a62b67.pdf (cit. on pp. 46, 56).

[52] J. A. Garay, A. Kiayias, N. Leonardos, G. Panagiotakos. "Bootstrapping the Blockchain, with Applications to Consensus and Fast PKI Setup". In: *IACR International Workshop on Public Key Cryptography*. Springer. 2018, pp. 465–495. DOI: 10.1007/978-3-319-76581-5_16 (cit. on pp. 44, 60).

[53] A. E. Gencer, S. Basu, I. Eyal, R. Van Renesse, E. G. Sirer. *Decentralization in bitcoin and ethereum networks*. 2018. arXiv: 1801.03998 [cs.CR] (cit. on pp. 43, 64).

[54] G. Greenspan. *Multichain private blockchain, white paper*. 2015. URL: http://www.multichain.com/download/MultiChain-White-Paper.pdf (cit. on pp. 49, 62, 63).

[55] *Guidelines For The Issuance And Management Of Extended Validation Certificates*. 2018. URL: https://cabforum.org/wp-content/uploads/CA-Browser-Forum-EV-Guidelines-v1.6.8.pdf (visited on 03/19/2019) (cit. on p. 25).

[56] N. Gura, A. Patel, A. Wander, H. Eberle, S. C. Shantz. "Comparing elliptic curve cryptography and RSA on 8-bit CPUs". In: *International workshop on cryptographic hardware and embedded systems*. Springer. 2004, pp. 119–132. DOI: 10.1007/978-3-540-28632-5_9 (cit. on p. 21).

[57] E. Heilman, A. Kendler, A. Zohar, S. Goldberg. “Eclipse attacks on bitcoin’s peer-to-peer network”. In: *24th {USENIX} Security Symposium ({USENIX} Security 15)*. 2015, pp. 129–144. ISBN: 978-1-931971-232. URL: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/heilman (cit. on pp. 43, 44).

[58] R. Holz, L. Braun, N. Kammenhuber, G. Carle. “The SSL landscape: a thorough analysis of the x. 509 PKI using active and passive measurements”. In: *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*. ACM. 2011, pp. 427–444. DOI: 10.1145/2068816.2068856 (cit. on p. 17).

[59] *HTTP Public Key Pinning (HPKP)*. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Public_Key_Pinning#Browser_compatibility (visited on 04/09/2019) (cit. on p. 31).

[60] International Energy Agency. *Electricity Statistics*. URL: https://www.iea.org/statistics/electricity/ (visited on 04/30/2019) (cit. on p. 39).

[61] Internet Security Research Group (ISRG). *Certification Practice Statement*. 2016. URL: https://letsencrypt.org/documents/ISRG-CPS-March-16-2016.pdf (visited on 03/11/2019) (cit. on p. 28).

[62] A. Judmayer, A. Zamyatin, N. Stifter, A. G. Voyiatzis, E. Weippl. *Merged Mining: Curse of Cure?* Cryptology ePrint Archive, Report 2017/791. https://eprint.iacr.org/2017/791. 2017 (cit. on p. 64).

[63] H. Kalodner, M. Carlsten, P. Ellenbogen, J. Bonneau, A. Narayanan. “An empirical study of Namecoin and lessons for decentralized namespace design”. In: *WEIS ’15: Proceedings of the 14th Workshop on the Economics of Information Security* (June 2015). URL: http://randomwalker.info/publications/namespaces.pdf (cit. on p. 69).

[64] T. H.-J. Kim, L.-S. Huang, A. Perrig, C. Jackson, V. Gligor. “Accountable key infrastructure (AKI): A proposal for a public-key validation infrastructure”. In: *Proceedings of the 22nd international conference on World Wide Web*. ACM. 2013, pp. 679–690. DOI: 10.1145/2488388.2488448 (cit. on pp. 18, 31).

[65] M. Y. Kubilay, M. S. Kiraz, H. A. Mantar. *CertLedger: A New PKI Model with Certificate Transparency Based on Blockchain*. 2018. arXiv: 1806.03914 [cs.CR] (cit. on p. 46).

[66] R. Küsters, T. Wilke. *Moderne Kryptographie*. Vieweg+Teubner, 2008. DOI: 10.1007/978-3-8348-8288-2 (cit. on pp. 22, 23, 28, 93).

[67] *Largest Crypto-Mining Marketplace*. URL: https://www.nicehash.com/ (visited on 04/13/2019) (cit. on pp. 41, 42).

[68] B. Laurie, E. Kasper. “Revocation transparency”. In: *Google Research, September* (2012). URL: https://www.links.org/files/RevocationTransparency.pdf (cit. on p. 33).

[69] B. Laurie, A. Langley, E. Kasper. *Certificate Transparency*. RFC 6962. June 2013. DOI: 10.17487/RFC6962 (cit. on pp. 18, 31, 95).

[70] B. Leiding, C. H. Cap, T. Mundt, S. Rashidibajgan. *Authcoin: validation and authentication in decentralized networks*. 2016. arXiv: 1609.04955 [cs.CR] (cit. on pp. 30, 46).

[71] *Let’s Encrypt Stats*. URL: https://letsencrypt.org/stats/ (visited on 04/10/2019) (cit. on p. 28).

[72] *List of available trusted root certificates in iOS 12, macOS 10.14, watchOS 5, and tvOS 12*. URL: https://support.apple.com/en-us/HT209144 (visited on 03/26/2019) (cit. on p. 27).

[73] Z. Lu, Q. Wang, G. Qu, Z. Liu. "Bars: a blockchain-based anonymous reputation system for trust management in vanets". In: *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE. 2018, pp. 98–103. DOI: 10.1109/trustcom/bigdatase.2018.00025 (cit. on p. 46).

[74] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, A. Hobor. *Making Smart Contracts Smarter*. Cryptology ePrint Archive, Report 2016/633. https://eprint.iacr.org/2016/633. 2016 (cit. on p. 79).

[75] Y. Marcus, E. Heilman, S. Goldberg. *Low-Resource Eclipse Attacks on Ethereum's Peer-to-Peer Network*. Cryptology ePrint Archive, Report 2018/236. https://eprint.iacr.org/2018/236. 2018 (cit. on p. 43).

[76] N. Marinoff. *Bitmain Nears 51% of Network Hash Rate: Why This Matters and Why It Doesn't*. 2018. URL: https://www.nasdaq.com/article/bitmain-nears-51-of-network-hash-rate-why-this-matters-and-why-it-doesnt-cm985195 (visited on 04/17/2019) (cit. on p. 43).

[77] S. Matsumoto, R. M. Reischuk. *IKP: Turning a PKI Around with Blockchains*. Cryptology ePrint Archive, Report 2016/1018. https://eprint.iacr.org/2016/1018. 2016 (cit. on pp. 46, 49, 71).

[78] N. Mcburnett. *PGP Web of Trust Statistics*. URL: http://bcn.boulder.co.us/~neal/pgpstat/19961206/ (visited on 04/19/2019) (cit. on p. 29).

[79] M. S. Melara, A. Blankstein, J. Bonneau, E. W. Felten, M. J. Freedman. "{CONIKS}: Bringing Key Transparency to End Users". In: *24th {USENIX} Security Symposium ({USENIX} Security 15)*. 2015, pp. 383–398. ISBN: 978-1-931971-232. URL: http://dl.acm.org/citation.cfm?id=2831143.2831168 (cit. on pp. 30, 68).

[80] S. Moosavi, J. Clark. "Ghazal: Toward Truly Authoritative Web Certificates Using Ethereum". In: *Financial Cryptography and Data Security*. Springer Berlin Heidelberg, 2019, pp. 352–366. DOI: 10.1007/978-3-662-58820-8_24 (cit. on pp. 46, 71).

[81] *MultiChain 1.0 beta 2 and 2.0 roadmap*. 2017. URL: https://www.multichain.com/blog/2017/06/multichain-1-beta-2-roadmap/ (visited on 04/20/2019) (cit. on p. 62).

[82] *Mythril Classic: Security analysis tool for Ethereum smart contracts*. URL: https://github.com/ConsenSys/mythril-classic (visited on 04/30/2019) (cit. on p. 79).

[83] S. Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*. 2008. URL: https://allquantor.at/blockchainbib/pdf/nakamoto2008bitcoin.pdf (cit. on pp. 18, 35, 38, 39, 41, 95).

[84] *Namecoin*. URL: https://namecoin.org/docs/faq/ (visited on 02/15/2019) (cit. on pp. 35, 55, 60, 69).

[85] M. Nystrom, B. Kaliski. *PKCS #10: Certification Request Syntax Specification Version 1.7*. RFC 2986. Nov. 2000. DOI: 10.17487/RFC2986 (cit. on p. 25).

[86] D. O'Brien, R. Sleevi, E. Stark. *Distrust of the Symantec PKI: Immediate action needed by site operators*. 2018. URL: https://security.googleblog.com/2018/03/distrust-of-symantec-pki-immediate.html (visited on 04/18/2019) (cit. on p. 18).

[87] H. P. Penning. *Analysis of the strong set in the PGP web of trust*. 2018. URL: https://pgp.cs.uu.nl/plot/ (visited on 12/10/2018) (cit. on p. 30).

[88] *PoW 51% Attack Cost*. URL: https://www.crypto51.app/ (visited on 04/13/2019) (cit. on pp. 41, 42).

[89] J. R. Prins, B. U. Cybercrime. "Diginotar certificate authority breach 'operation black tulip'". In: *Fox-IT, November* (2011). URL: https://www.teletrust.de/uploads/media/FOX-IT_Interim_Report_2011-09-05_pdf.pdf (cit. on p. 18).

[90] *Public Key Service*. URL: https://www.telesec.de/en/signaturecard (visited on 04/19/2019) (cit. on p. 23).

[91] B. Qin, J. Huang, Q. Wang, X. Luo, B. Liang, W. Shi. "Cecoin: A decentralized PKI mitigating MitM attacks". In: *Future Generation Computer Systems* (2017). DOI: 10.1016/j.future.2017.08.025 (cit. on p. 46).

[92] *Remix - Solidity IDE*. URL: https://remix.ethereum.org (visited on 04/29/2019) (cit. on pp. 65, 77).

[93] S. S. Al-Riyami, K. G. Paterson. "Certificateless public key cryptography". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2003, pp. 452–473. DOI: 10.1007/978-3-540-40061-5_29 (cit. on p. 17).

[94] *Root Certificate Policy*. URL: https://www.chromium.org/Home/chromium-security/root-ca-policy (visited on 03/26/2019) (cit. on p. 27).

[95] S. Ruoti, J. Andersen, D. Zappala, K. Seamons. *Why Johnny still, still can't encrypt: Evaluating the usability of a modern PGP client*. 2015. arXiv: 1510.08555 [cs.CR] (cit. on p. 29).

[96] M. D. Ryan. "Enhanced Certificate Transparency and End-to-End Encrypted Mail." In: *NDSS*. 2014. DOI: 10.14722/ndss.2014.23379 (cit. on p. 33).

[97] D. Rystsov. *Nidaba: a distributed scalable PKI with a stable price for certificate operations*. 2014. URL: http://rystsov.info/files/nidaba.pdf (cit. on p. 46).

[98] *SafeMath*. URL: https://docs.openzeppelin.org/docs/math_safemath (visited on 04/30/2019) (cit. on p. 79).

[99] A. Samoshkin. *SSL certificate revocation and how it is broken in practice*. 2018. URL: https://medium.com/@alexeysamoshkin/how-ssl-certificate-revocation-is-broken-in-practice-af3b63b9cb3 (visited on 04/26/2019) (cit. on p. 27).

[100] S. Santesson, M. Myers, R. Ankney, A. Malpani, S. Galperin, D. C. Adams. *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*. RFC 6960. June 2013. DOI: 10.17487/RFC6960 (cit. on p. 25).

[101] S. Sheng, L. Broderick, C. A. Koranda, J. J. Hyland. "Why johnny still can't encrypt: evaluating the usability of email encryption software". In: *Symposium On Usable Privacy and Security*. 2006, pp. 3–4. URL: https://cups.cs.cmu.edu/soups/2006/posters/sheng-poster_abstract.pdf (cit. on p. 29).

[102] K. Sirota. *Trusted Root Program Participants (as of November 27 2018)*. 2018. URL: https://gallery.technet.microsoft.com/Trusted-Root-Program-831324c6 (visited on 03/26/2019) (cit. on p. 27).

[103] R. Sleevi. *Sustaining Digital Certificate Security*. 2015. URL: https://security.googleblog.com/2015/10/sustaining-digital-certificate-security.html (visited on 03/11/2019) (cit. on p. 18).

[104] *SSL-Zertifikate*. URL: https://de.godaddy.com/web-sicherheit/ssl-zertifikat (visited on 04/19/2019) (cit. on p. 23).

[105] E. Stark, R. Sleevi, R. Muminović, D. O'Brien, E. Messeri, A. P. Felt, B. McMillion, P. Tabriz. "Does Certificate Transparency Break the Web? Measuring Adoption and Error Rate". In: 2019. URL: https://ai.google/research/pubs/pub47551 (cit. on pp. 25, 32).

[106] StatCounter. *Global market share held by leading desktop internet browsers from January 2015 to December 2018*. URL: https://www.statista.com/statistics/544400/market-share-of-internet-browsers-desktop/ (visited on 04/16/2019) (cit. on p. 26).

[107] D. Sui, S. Ricci, J. Pfeffer. *Are Miners Centralized? A Look into Mining Pools*. 2018. URL: https://media.consensys.net/are-miners-centralized-a-look-into-mining-pools-b594425411dc (visited on 04/18/2019) (cit. on p. 64).

[108] M. Szydlo. "Merkle tree traversal in log space and time". In: *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2004, pp. 541–554. DOI: 10.1007/978-3-540-24676-3_32 (cit. on p. 94).

[109] D. Taylor. "An Analysis of Bitcoin and the Proof of Work Protocols Energy Consumption, Growth, Impact and Sustainability". MA thesis. University of Strathclyde Glasgow, 2018. URL: http://www.esru.strath.ac.uk/Documents/MSc_2018/Taylor.pdf (cit. on p. 39).

[110] H. Tewari, A. Hughes, S. Weber, T. Barry. "X509Cloud—Framework for a ubiquitous PKI". In: *Military Communications Conference (MILCOM), MILCOM 2017-2017 IEEE*. IEEE. 2017, pp. 225–230. DOI: 10.1109/milcom.2017.8170796 (cit. on pp. 46, 48).

[111] A. Tomescu, S. Devadas. "Catena: Efficient non-equivocation via bitcoin". In: *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2017, pp. 393–409. DOI: 10.1109/sp.2017.19 (cit. on pp. 46, 54, 62).

[112] *TransparencyReport: HTTPS encryption on the web*. 2019. URL: https://transparencyreport.google.com/https/certificates?hl=en (visited on 02/01/2019) (cit. on p. 32).

[113] *Truffle Suite*. URL: https://truffleframework.com/ (visited on 04/30/2019) (cit. on p. 77).

[114] *Use of DNSSEC Validation for World (XA)*. URL: https://stats.labs.apnic.net/dnssec/XA?o=cXAw30x1g1r1 (visited on 04/10/2019) (cit. on p. 33).

[115] N. H. Walfield. *An Advanced Introduction to GnuPG*. 2017. URL: https://gnupg.org/ftp/people/neal/an-advanced-introduction-to-gnupg/an-advanced-introduction-to-gnupg.pdf (cit. on pp. 28, 30).

[116] H. Wang, Z. Zheng, S. Xie, H. N. Dai, X. Chen. "Blockchain challenges and opportunities: a survey". In: *International Journal of Web and Grid Services* 14.4 (2018), p. 352. DOI: 10.1504/ijwgs.2018.10016848 (cit. on p. 39).

[117] A. Whalley. *Distrusting WoSign and StartCom Certificates*. 2016. URL: https://security.googleblog.com/2016/10/distrusting-wosign-and-startcom.html (visited on 03/11/2019) (cit. on p. 18).

[118] A. Whitten, J. D. Tygar. "Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0." In: *USENIX Security Symposium*. Vol. 348. 1999. URL: https://users.ece.cmu.edu/~adrian/630-f05/readings/whitten-tygar.pdf (cit. on p. 29).

[119] Z. Wilcox-O'Hearn. *Names: Distributed, Secure, Human-Readable: Choose Two*. 2001. URL: https://web.archive.org/web/20011020191610/http://zooko.com/distnames.html (visited on 03/15/2019) (cit. on p. 95).

[120] D. Wilson, G. Ateniese. "From pretty good to great: Enhancing PGP using bitcoin and the blockchain". In: *International conference on network and system security*. Springer. 2015, pp. 368–375. DOI: 10.1007/978-3-319-25645-0_25 (cit. on pp. 28, 46, 74).

[121] *WISeKey announces definitive agreement to sell QuoVadis to DigiCert*. 2018. URL: https://www.quovadisglobal.co.uk/Corporate/NewsAndEvents/20181224_Digicert_Acquires_QuoVadis.aspx (visited on 01/08/2019) (cit. on p. 17).

[122] G. Wood. *Ethereum: A secure decentralised generalised transaction ledger. Ethereum Project Yellow Paper (2014)*. BYZANTIUM VERSION 2d0661f - 2018-11-08. 2014 (cit. on pp. 35–37, 95).

[123] A. Yakubov, W. M. Shbair, A. Wallbom, D. Sanda, R. State. "A blockchain-based PKI management framework". In: *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2018. DOI: 10.1109/noms.2018.8406325 (cit. on p. 46).

[124] A. Yakubov, W. Shbair, R. State. "BlockPGP: A Blockchain-Based Framework for PGP Key Servers". In: *2018 Sixth International Symposium on Computing and Networking Workshops (CANDARW)*. IEEE. 2018, pp. 316–322. DOI: 10.1109/candarw.2018.00065 (cit. on pp. 46, 53, 67).

All links were last followed on May 2, 2018.

A Appendix

A.1 Hash Functions

Hash functions are functions that take a message of any length as an input, and output a bit string of limited length. Output of hash function is called a hash or digest. It can be defined as follows [66]:

$$h : \{0,1\}^* \longrightarrow \{0,1\}^l$$
$$x \longmapsto y$$

where $\{0,1\}^*$ is a set of bit strings of any length, $l \in \mathbb{N}$, and $\{0,1\}^l$ a set of bit strings of length l.

In information security hash functions are expected to be collision resistant which means that an attacker should not be able to find any pair of messages x_1 and x_2 such that $h(x_1) = h(x_2)$. Examples of hash functions used in practice are SHA-2, SHA-3, and BLAKE.

Hash functions are used as a building block of cryptographic primitives such as message authentication codes (MACs), digital signatures and other more complicated structures like Merkle trees.

A.2 Digital Signatures

Digital signature is a cryptographic primitive that provides integrity and non-repudiation. It relies on the notion of public and private key from public key cryptography. Digital signature provides security in the scenario where Alice wants to be sure that the message which she sends to Bob is not tampered during transfer. Hence, she executes a signing algorithm using her private key and the message as inputs. She includes the output of this operation into message before sending it to Bob.

More formally, according to [66], digital signatures can be defined as a tuple (X, G, K, T, V) where

- $X \subseteq \{0,1\}^*$ is a set of plaintexts
- G is a probabilistic key generation function which generates $(k, \hat{k})$, a pair containing a public key k and private key $\hat{k}$
- $K = K_{pub} \times K_{priv}$ is a set of key pairs such that $(k, \hat{k}) \in K$
- T is a probabilistic signing algorithm which given input $x \in X$ and $\hat{k} \in K_{priv}$ outputs a signature $s \in \{0,1\}^*$
- V is a deterministic verification algorithm which given a plaintext $x \in X$, signature $s \in \{0,1\}^*$ and public key $k \in K_{pub}$ outputs 0 or 1

and the perfect correctness property (A.1) holds:

$$\forall x \in X, (k, \hat{k}) \in K : V(x, T(x, \hat{k}), k) = valid \tag{A.1}$$

Non-repudiation property arises from the fact that Alice signs messages with her private key which is only known to her. Thus, when Bob receives a message signed by Alice, he can be sure that the message was sent by Alice if the verification function V outputs *valid* when he runs it. As always in public key cryptography, digital signatures rely on the fact that Bob is assured the corresponding public key indeed belongs to Alice.

Digital signatures are used in many areas of information science: in blockchain user account is represented through a public and private key allowing every transaction sent by the user to be signed using her private key; in CA-based PKIs CA uses its private key to sign certificate, whereas clients in HTTPS connections verify the correctness of signatures with widely promulgated public keys of the CAs; in PGP Alice signs public keys of other users whom she trusts.

A.3 Merkle Tree

Merkle tree allows the creation of storage that provides users with the ability to check it for any changes. It assures the consistency of stored data through links between entries in the storage and their hashes which represent Merkle tree's leaves. Every two neighboring nodes from leaves to the root are thereafter concatenated, and the hash of every concatenation becomes a parent node. Since the root node depends on the hashes of every node down to the leaves, every addition, modification, or removal of leaves changes the root hash value.

According to [108], Merkle tree is a complete binary tree. Nodes in a Merkle tree consist of hashes. Depending on the position of node in a tree, the way its hash was calculated, and its significance, it is possible to differentiate between three types of nodes in a Merkle tree: (1) leaf node, (2) intermediate node, and (3) root node. Leaf node is created by hashing some value that can, for example, represent an entry in a log. This process can be seen more clearly if we let $x_1, x_2, ..., x_n$ ($n \in \mathbf{N}$) be a set of entries in the storage. Leaves of the Merkle tree are then created by hashing every entry (A.2).

$$\forall i \in 1, ..., n : leaf_i = h(x_i) \tag{A.2}$$

Another sort of nodes are intermediate nodes. Intermediate node on a level (distance from the root node) l is created by hashing the result of concatenation of its child nodes. Since Merkle tree is a complete tree, this kind of node can arise in three ways. If we assume that n is the bottom level of the tree, an intermediate node on level $n-1$ is usually created by hashing a concatenation of two neighboring leaf nodes (A.3). On the other hand, intermediate nodes on other levels can contain a hash of concatenation of an intermediate node and a leaf node (A.4), or a hash of two other intermediate nodes, $N1_{l+1}^{int}$ and $N2_{l+1}^{int}$ (A.5). Finally, on level 0 of the tree is the root node, which depending on the height of a tree can be generated as any other intermediate node. In the following equations || represents concatenation.

$$intermediate_{n-1} = h(h(x_i)||h(x_{i+1})) \tag{A.3}$$

$$intermediate_l = h(h(x_i)||intermediate_{l+1}) \tag{A.4}$$

$$intermediate_l = h(N1^{int}_{l+1}||N2^{int}_{l+1})) \tag{A.5}$$

Merkle tree has several positive features. Since it is a binary tree, it is efficient and lookup of every entry in the tree has a logarithmic complexity. Furthermore, any kind of modification to an entry results in the change of hash in the root node. This provides proof of consistency because it makes it possible to easily check whether content changed. Proof of entry inclusion can also be done with a set of intermediate nodes that occur on the path from the leaf node corresponding to an entry to the root.

These features make Merkle tree structure widely used in PKIs based on blockchains. Merkle tree and its variants are used in PKIs such as CONIKS, PKI extensions such as CT [69], and in blockchains [83], [122]. In blockchain, transactions are stored using a Merkle tree, whereas in CT it is used to store certificates in a log.

A.4 Zooko's Triangle

Zooko's triangle describes properties that are desirable for any identity system with name-value pairs that runs on a network [119]. It has been deemed that a single system cannot fulfill all of the properties. These properties are:

1. **Human-readable** Names should be meaningful for humans, e.g. data's hash is inherently associated with data, but it is usually not meaningful to humans.
2. **Distributed** There is no trusted third party which controls a namespace.
3. **Secure** Adversary cannot change associations between names and data. This makes her unable to mount an impersonation attack.

A.5 IPFS

IPFS is a distributed file system in which network nodes communicate with each other through peer-to-peer networking [10]. Data is stored in IPFS in form of blocks that are exchanged between the nodes in IPFS network. Hashes of stored files are used as links to files. IPFS also has the ability to track versions of stored files, and supports large files which are then split into independent blocks.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature