International Journal of Networking and Computing – www.ijnc.org, ISSN 2185-2847
Volume 10, Number 1, pages 1–24, January 2020

BlockPGP: A Blockchain-based Framework for PGP Key Servers

Alexander Yakubov, Wazen Shbair, Nida Khan, Radu State
University of Luxembourg, SnT, 29, Avenue J.F Kennedy, L-1855 Luxembourg
{alexander.yakubov, wazen.shbair, nida.khan, radu.state}@uni.lu

Christophe Medinger, Jean Hilger
BCEE, 16, Rue Sainte Zithe, L-2763 Luxembourg
{c.medinger, j.hilger}@bcee.lu

Received: February 14, 2019
Revised: April 23, 2019
Revised: July 25, 2019
Accepted: September 2, 2019
Communicated by Yasuyuki Nogami

Abstract

Pretty Good Privacy (PGP) is one of the most prominent cryptographic standards offering end-to-end encryption for email messages and other sensitive information exchange. PGP allows to verify the identity of the correspondent in information exchange as well as the information integrity. PGP implements asymmetric encryption with certificates shared through a network of PGP key servers. In this paper, we propose a new PGP management framework with the key servers infrastructure implemented using blockchain technology. Our approach offers fast propagation of certificate revocation among PGP key servers and elimination of man-in-the-middle risks. It also grants users the required access control to update their own PGP certificates, which is not the case with the current PGP key servers. A prototype has been implemented using Ethereum blockchain and an open source key server, named Hockeypuck. Finally, we evaluated the prototype with extensive experiments. Our results show that our solution is practical and it could be integrated with the existing public PGP key servers infrastructure.

Keywords: PGP, Blockchain, Ethereum, Key server, PKI, Certificate Transparency, Web of Trust

1 Introduction

Pretty Good Privacy (PGP) is a popular cryptographic protocol used for encryption of e-mail messages, text and files, directories and even disk partitions. PGP concept is based on asymmetric public key cryptography for user identification, where each user is given public and private keys. To send confidential information, the sender firstly retrieves the recipient public key and then the sensitive information (e.g. email message) is encrypted with temporary symmetric key. Thereafter, the sender encrypts the symmetric key with the receiver's public key and adds the encrypted symmetric key to the message. As the public key is derived from the private key, but not vice versa, only the private key holder is able to decrypt and read the message [4]. The distribution and management of PGP key pair is one of the main issues that needs to be solved.

In contrast to traditional centralized hierarchical Public Key Infrastructure (PKI) used for issuing SSL/TLS certificate by Certificate Authority (CA), PGP adopts distributed trust network for

certificate identity confirmation known as "Web of Trust". In PGP there is no central authority which everybody trusts, but instead, participants sign each other's keys and progressively build a web of individual public keys interconnected by links formed by these signatures.

Despite popularity of SSL/TLS certificates we believe PGP is likely to benefit from the growth of its market share in coming years due to extensive introduction of private blockchains. Indeed, presently user identification is often supported by SSL-based PKI solutions like certificate authority membership services[1] at HyperLedger Fabric due to simplicity of traditional PKI. However, SSL-based PKI with its centralized client-server architecture contradicts distributed concept of blockchain. With the launch of private blockchain applications, identification services shall be provided based on decentralized solutions, and we do not see an alternative here to Web of Trust framework implemented in PGP. Hence, in our view PGP key infrastructure is overlooked by research community which presently focuses on more popular SSL-based PKI.

Although PGP benefits from the absence of centralized CAs being the points of failure in PKI due to PGP's distributed architecture, the validity assessment of certification paths can become expensive with the growth of participant amount (graph vertices) [9]. As one of solutions to handle this issue extensive key management infrastructure was built based on PGP key servers supported by the community. Until recently PGP key infrastructure boasted some advantages compared to PKI suffering from its centralization and inefficient revocations. However, following the implementation of PKI log-based infrastructure solutions like Google's Certificate Transparency, the security challenges of PGP key servers became apparent, including MITM risks and delays with certificate revocation. According to description of OpenPGP HTTP Keyserver Protocol (HKP)[2], PGP key servers presently are not considered as a secure means of public key distribution.

In this paper we will discuss how to solve security challenges of PGP key servers using blockchain technology.

The contribution and novelty of this paper is threefold:

1. We designed a blockchain-based framework (BlockPGP) providing reliable management for PGP certificates and key server infrastructure. Based on implementation of this framework, PGP key server security challenges, including MITM risks and insufficient revocation efficiency, can be solved.

2. We proposed a technique of blockchain-related data incorporation into PGP certificate in line with the current OpenPGP standard. Integration of blockchain-related information into PGP certificate is crucial for arrangement of user access control in certificate management infrastructure.

3. We developed a prototype that demonstrates the practical applicability of the proposed approach. The prototype is available at ***https://github.com/alyakubov/blockpgp*** as an open source project. It includes a stand-alone Unix application, providing the access to blockchain-based key infrastructure, as well as traditional PGP key server, based on open source Hockeypuck software[3] for synchronization of blockchain key storage with existing PGP key management infrastructure.

Outline. The rest of the paper is organized as follows: Section 2 provides a background of the PGP distributed trust model and its differences from the traditional PKI hierarchical trust concept. In Section 3 we discuss the security challenges of PGP key servers as well as a range of potential solutions, including those based on blockchain technology. Section 4 explores the work related to blockchain-based certificate management, concept of distributed identity storage to be implemented in some products like Hyperledger Indy, etc. In Section 5 we will provide some details on deployment of our blockchain-based solution and discuss its performance. Finally, Section 6 concludes the paper.

[1] https://hyperledger-fabric.readthedocs.io/en/release-1.4/msp.html
[2] https://tools.ietf.org/html/draft-shaw-openpgp-hkp-00
[3] https://hockeypuck.github.io

2 Background and Preliminaries

In this section we present the concept of trust in PGP (distributed Web of Trust) and PKI (hierarchical centralized system), then we will review some approaches to trust assessment in PGP and finally we will discuss the security issues in PGP and PKI protocols.

2.1 Web of Trust

Web of Trust is decentralized public key environment where each participant of the ecosystem can "introduce" (sign) public keys of other participants. It substantially differs from the centralized hierarchical concept of traditional SSL/TLS certificates, also known as Public Key Infrastructure (PKI) discussed below. In PKI a certificate can be introduced only by Certificate Authority (CA), a participant with a special status, while in PGP any participant can be considered as a certificate authority from PKI viewpoint.

PGP users are able to designate public keys of others with different levels of trust indicating how trustworthy the signature (introduction) of the certificate holder is, when she or he signs public key certificates of other participant. In other words, how trustworthy her/his introduction of other participants is. There are four levels of trust in PGP regarding introduction of other certificates:

- ***full* (level=4)**: certificate holder's signature of other users' certificates is fully trusted
- ***marginal* (level=3)**: certificate holder's signature can be trusted, but it is better to find other signatures with full trust to confirm the introduction of a certain certificate
- ***untrustworthy* (level=2)**: certificate holder's signature should not be trusted and his signature on other users' certificates should be ignored
- ***don't know* (level=1)**: there is uncertainty about trustworthiness of certificate holder's signatures of other users' certificates

Based on multiple introductions (signatures) of PGP certificates by different users it is possible to create a "chain of trust", or a path from one user to another when identity confirmation is required for sending protected information. Basically this problem can be considered as finding paths between two vertices in a directed graph with variable vertex weights. To reach a reasonable identification trust level for a given third party public key, one should find a path with number of edges less than six ([1], Chapter 3). Notably, trust levels for signatures, mentioned above, are stored locally in GNU OpenPGP application's data (*gpg* utility) and, to the best of our knowledge, are not loaded to PGP key servers discussed below. In the framework of our Proof-of-Concept, we will not use signature trust level data of PGP participants.

The main advantage of PGP's Web of Trust public key infrastructure stems from its distributed nature as it excludes any central point of failure. However, this implies some difficulty for new or remote users to join the network, since some existing member of Web of Trust typically must meet with the new user in person to have her/his identity verified and public key signed for the first time (usually it is done on PGP *Keysigning events*[4] arranged for authentication of new participants).

2.2 Public Key Infrastructure (PKI)

In contrast to PGP with its distributed introduction of certificates known as Web of Trust, Public Key Infrastructure (PKI) is a hierarchical structure of certificate issuers (Certificate Authorities or CAs) authenticating identities over Internet. The PKI defines the certificate issue and management policies which for instance envisage periodic audits of Certificate Authorities as well as other security measures [27]. PKI management of public keys is based on the certificate standard X.509 defined as a data structure that binds public key values to subjects (for instance, domain names). The binding is asserted by some Certificate Authorities (CA) digitally signing each certificate. The CA may base this assertion on profound validation of the private certificate holder's identity [11].

[4] https://www.debian.org/events/keysigning

Table 1: The distribution of PGP certificates based on the number of signers (on 15.11.2018)

No. of Signers	No. of PGP Keys	Percentage	
0	4394932	84.23%	84.23%
1	424815	8.14%	
2	131911	2.53%	
3	58944	1.13%	
4	37882	0.73%	
5	20632	0.40%	
6	18302	0.35%	15.77%
7	12039	0.23%	
8	11514	0.22%	
9	9150	0.18%	
10	7933	0.15%	
more than 10	89420	1.71%	

There are three types of X.509 certificates according to the depth of verification. The most simple one is a Domain Validated certificate (DV), it asserts that a domain name is mapped to the correct web server (IP address) through DNS. The Organization Validated certificate (OV) also envisages additional CA-verified information, such as an organization name and a postal address. These extra validations make OV SSL Certificates more expensive compared to DV certificates. Extended Validation certificate (EV) implies the highest level of authentication, including diligent human validation of the site's identity and business registration details. Because of the extensive validation, EV is the most expensive among digital certificates [3].

2.3 Concept of trust assessment in PGP

On January 5th, 2019 PGP key servers held approximately 5.4 million certificates. The exact number of certificates varies from one key server to another due to synchronization delays, we will discuss this issue below. As mentioned above, any holder of PGP certificate can sign any certificate, implying the confirmation of the owner identity of the signed certificate and thus forming a Web of Trust. However, only around 16% of PGP certificates held in the storage of PGP servers are signed by other PGP certificate holders (shown in Table 1). From this point of view, it is more correctly to use native PGP terminology referring to a PGP public key distributed with key servers just as a key (or signed key) rather than certificate. Indeed, theoretically speaking only a public key signed by a third party can be considered as a certificate. Nevertheless we use in this work PKI terminology for consistency, where we refer to PGP public keys as certificates or self-signed certificates if they are not signed by a third party.

Table 1 shows that only 8% of total PGP certificates held by key servers have more than one signature. Given the portion of signed PGP certificates of 16%, half of all certificates signed by third parties have only one signature. As we discussed before, PGP certificate can be signed by any third party certificate holder rather than pre-approved authorities like PKI CAs, meaning that one should not trust PGP certificates with just one signature due to potential private key breach.

But how many signatures should PGP certificate have to be trusted? There are many different views on the subject as two signatures may still be rather weak identity confirmation, given that ordinary PGP users are not subjected to technical audit in contrast to PKI CAs and it is not that difficult to breach private keys of two PGP users. Certificates with more than two signatures account only 5% of total number of certificates loaded to PGP key server storage (Figure 1). The idea of PGP trust concept is based on assumption that each end-user makes his own decision how many signatures are necessary and which algorithm to use for trust assessment.

Another approach to trust assessment described in numerous PGP best practice guides is based on the directed graph path of signatures. If there is a path connecting certificate in question and the user's own certificate then it is assumed that the certificate can be trusted. An additional condition can be introduced regarding path length (minimum number of edges) not exceeding predefined

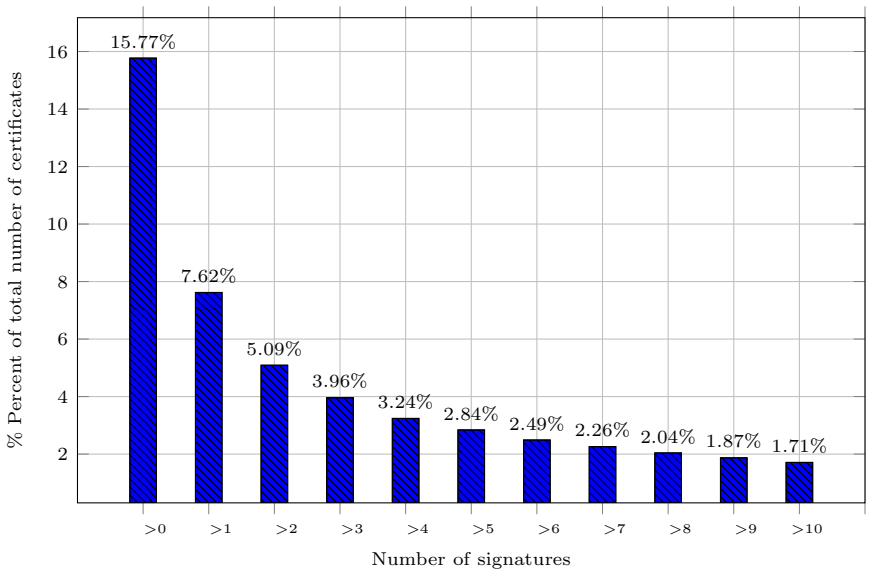

Figure 1: Percentage of PGP certificates with third-party signatures greater than the specified amount (on 15.11.2018)

number N (often $N = 5$). On the other hand the PGP trust graph's strong set comes to 60,000 certificates accounting for around 1.1% of the total number of certificates in key server storage[5]. Strong set is the largest set of certificates such that for any two certificates in the set there is a directed path from one to the other. Notably the most probable path length in the strong set comes to around 5.7 which corresponds to best practices' recommended trust path length of no more than 5 edges (for instance, [1], Chapter 3). As the share of strong set in total amount of certificates in PGP key servers' storage is 1.6 times lower compared to the share of certificates with more than 10 signatures, the trust assessments based on certification (signature) paths might be overly conservative.

2.4 Security issues of PGP and PKI

Given its more flexible trust model based on Web of Trust concept, PGP should be advantageous from security view point compared to PKI with its potential breaches of Certificate Authorities (CAs) being PKI's central point of failure.

However, recent research [22] disclosed a major breach in some mail clients with PGP support allowing to retrieve decrypted body of a protected email message with manipulations of HTML tags built-in to the message. Although this vulnerability can be resolved by careful use of the client software with PGP support, still there are many questions regarding PGP security. For instance, there are potential problems with PGP key servers including Man-in-the-Middle (MITM) attack. For example, this MITM attack may stem from potential breaches of PKI used in users' exchange with PGP key servers. Details regarding PGP key servers' security problems will be discussed below.

The security of PKI systems relies on infrastructure of Certificate Authorities (CAs). To resolve CAs' risks as single centers of failure, it is possible to impose strict technical and management requirements (for instance, the audits we discussed above). However, many breaches show that the security of CAs can be compromised or subjected to operational errors. As CAs are designed to be crucial for PKI's security, CAs' errors or mismanagement have resulted in unauthorized certificates being issued [6]. For instance, one of well known events is the security breach of CA DigiNotar, which led to use of the company's infrastructure for the issue of hundreds of rogue digital certificates

[5]https://pgp.cs.uu.nl/plot/

for high-profile domains, including those for Google.com and Facebook.com. These certificates were later used in a mass surveillance attack against some Internet users [2].

In another PKI breach, a Malaysian subordinate certificate authority DigiCert Sendirian Berhad (DigiCert Sdn. Bhd.) mistakenly issued 22 weak SSL certificates, which could be used to impersonate websites and sign malicious software. As a result, major browsers had to revoke their trust in all certificates issued by DigiCert Sdn. Bhd. (Note: DigiCert Sdn. Bhd. is not affiliated with the U.S.-based corporation DigiCert, Inc.) [2].

Relatively recently another type of PKI certificate mismanagement was found in two independent cases[6] of French certificate authority ANSSI (2013) and Chinese certificate authority CNNIC/MCS Holding (2015). For example, CNNIC issued a sub-CA certificate to MCS Holding to manage the domains and sites MCS has registered, but MCS put its CA certificate to its corporate proxy thus allowing users behind the proxy to issue certificates beyond the control of MCS CA administrators.

The above breach events illustrate how easy it is for CAs to make operational and management errors. The consequences of CA breaches can be quite severe. Additionally, revocation of compromised PKI certificates is also a challenge as each CA has its own Certificate Revocation List (CRL) and by default there is no fast procedure in place to check the validity of a given certificate. Importantly, CRLs lack efficiency due to delays in publication of updated revocation lists as well as the absence of universal access point to all CRLs. Recent initiatives like public PKI logs provide some solutions to existing certificate revocation challenges.

Log-based PKI approach has been proposed as a technique for fast and efficient revocation of the certificates, hence reducing the risks stemming from PKI certificate compromises. The idea behind is using highly-available public log servers that monitor and publish the certificates issued by CAs. These public logs provide transparency by ensuring that only publicly-logged certificates are accepted and trusted by end-customers, thus any CAs' misbehavior will be detected. Importantly, PGP historically had own analogue of log-based key publication, i.e. PGP key servers which we will discuss in details in the next section.

Google's Certificate Transparency (CT) [18] is the most widely deployed log-based PKI, and it is currently available in both Chrome and Firefox web browsers. In parallel, many proposals intend to extend the features of log-based PKI schema with further support for revocation and error handling. Unfortunately, despite these benefits, log-based PKIs still have several challenges related to the deployment process as explained in [20, 21]. One of the risks still remaining in the CT and other log-based PKIs is MITM risk [10].

PKI security issues are also crucial for PGP as, for instance, the present exchange among key servers and end-customers is often conducted with SSL/TLS protocol using PKI certificates. Moreover, we will show in this work that PGP key servers are exposed to MITM risk even at greater extend compared to log-based PKIs such as Certificate Transparency.

3 Security Challenges of PGP Key Servers

In this section, first we discuss PGP key servers and the security issues they have, namely MITM risks and inefficient revocation due to synchronization delays. Then we look at existing solutions to handle certificate revocation challenges like Goggle's Certificate Transparency. Finally, we will review blockchain as a technology capable of solving core security challenges of PGP key servers based on its built-in functionality.

3.1 PGP key servers

PGP key servers are an important component in OpenPGP concept. Participants of PGP environment are supposed to upload to these servers their certificate updates including signatures of other participants, expiration dates, revocations, etc. If the sender does not have public key of the receiver on her/his local computer she/he may also download it from a key server.

[6] https://security.googleblog.com/2015/03/maintaining-digital-certificate-security.html

Presently there are about 150 PGP key servers, but substantial portion of them are synchronized rather periodically and can be inactive. Key servers are gossiping to each other based on SKS protocol. According to statistics[7] on January 20, 2019 the primary SKS pools included around 60 key servers. Apart from the primary pools, there are also about 10-20 servers with relatively up-to-date synchronization.

However if one wants to use PGP key servers infrastructure it is important to bear in mind the following:

- ***PGP key servers should become more trustworthy***: according to Security section in OpenPGP HTTP Keyserver Protocol description[8], PGP key servers should not be trusted and must be used for informational purposes only. Basically it means that key servers do not provide full user access control and an attacker can upload compromised certificate of other users to the key server's storage. Blockchain-based certificate storage, which we will discuss in this work, grants write access only to the certificate holder's Ethereum account specified in the certificate (and also to the administrator).

- ***Man-in-the-middle risk***: end-user interaction with PGP key servers is often conducted with HTTP Key Server Protocol (HKP), which can be protected with SSL/TLS protocol. Given the potential vulnerabilities of PKI stemming from its single point of failure at CAs, it can be concluded that the whole PGP ecosystem becomes dependent on PKI security. Given breaches of CA security, connection of an end-user to PGP key server can be exposed to MITM risks. On the other hand, due to extensive number of inactive PGP key servers the attacker can get control of one of them and manipulate user request implying the potential MITM attack. We will discuss in details how MITM attack to PGP end-user can be arranged below as well as how these risks can be eliminated with blockchain.

- ***Key server synchronization delays***: PGP key servers provide important functionality like unified publication of certificate revocations, which until recently was not fully available in PKI environment. However the delays with PGP key server synchronizations, which may take 15-30 hours as will be shown below, can result in security breaches with compromised certificates already revoked by their owners. We will discuss the synchronization delays of PGP key servers in section 3.3.

- ***Some key servers are not active***: as we discussed above up to half of PGP servers on the radar of SKS protocol scanning software[7] do not synchronize very often (the delays can exceed several days and even more) implying a risk of downloading compromised or revoked certificate as a valid one. As presently end-user has no direct access to PGP key server storage (with blockchain implementation as distributed key storage such access can be easily arranged) the user has to check carefully whether she or he connects to active PGP key server with permanent synchronization.

3.2 Man-In-The-Middle risk of PGP key servers

MITM risk remains one of the core security challenges for modern PKI log servers like Google's Certificate Transparency as well as for PGP key servers which historically, to certain extent, provided logging functionality for PGP users.

Obviously PGP users in general and PGP key servers in particular are exposed to straightforward MITM risks based on compromised SSL certificates. As we mentioned above PGP servers often use SSL/TLS connection in information exchange with end-users which can be breached with rogue PKI certificate. However, due to the introduction of new techniques aimed at efficient and fast certificate revocation the risks stemming from SSL certificate breaches will be reduced.

We will not focus on potential MITM risks stemming from compromised SSL certificates provided by an attacker. As we discussed above, the problem of PKI CA breaches appears difficult to resolve

[7] https://sks-keyservers.net/status/
[8] https://tools.ietf.org/html/draft-shaw-openpgp-hkp-00

despite CAs' periodic audits and other management measures. In contrast to PKI with CAs being single point of failure, PGP assumes that it is user's responsibility to decide if the certificate is valid based on the signatures of other users (i.e. participants of Web of Trust). One has to take into account though that it is much easier to compromise private keys of ordinary PGP users compared to the potential breach of PKI CA exposed to technical audit. Thus, a PGP certificate with just a single signature from a third party is very unlikely can be fully trusted.

In this work, we will concentrate on the other kind of MITM risks based on the attacker's control of one of the nodes of the certificate services infrastructure. This kind of MITM risk is referenced as "split-world" risk and was discussed in details in [10] in relation to Certificate Transparency, but this concept can be applied to PGP key servers as well.

The idea behind split-world attack is the following:

- The attacker controls one of PGP key servers. Technically it can be done even easier than acquiring control of one of PKI certificate logs (a node of distributed logging) in Certificate Transparency. As we discussed, around half of all PGP key servers are not active and are updated only periodically.

- Attacker provides services to a predefined user (a victim) based on the fake copy of the certificate storage. In other words, the attacker provides fake certificates of third parties to the victim.

- On the other hand, during gossiping sessions with other PGP key servers based on SKS protocol, the attacker conducts all information exchange using the correct version of the certificate storage.

We will show in this work that blockchain-based techniques can resolve the risk of split-world attack as well as other challenges of PGP key servers, including delays of certificate spreading among key servers.

3.3 Why delays in PGP key servers' synchronization are crucial for revocations

Revocation is an important part in certificate management as any certificate can be potentially compromised and there should be an efficient procedure in place to quickly suspend the certificate validity. However, there are significant differences between PGP and PKI in the revocation approach.

In PGP revocation can be conducted by a "revocator", a PGP participant holding revocation key for a given certificate. The idea here can be connected with the risk of operating system breach for a PGP certificate holder as she/he may not be able to revoke the certificate himself. The certificate updates related to its revocation should be uploaded to PGP key server to stop downloading of this certificate for encryption of sensitive information.

In PKI the certificate can be revoked by the CA that issued this certificate. The certificate is then supposed to be published in certificate revocation list (CRL) of this CA. However, as there is no universal revocation list available, the check of certificate validity in PKI is not straightforward. Based on X.509 concept [11], certificate validation should be conducted along "Chain of Trust", the path from a given certificate up to its issuing CA, then to parent CA and so on, climbing the hierarchy up to the Root CA certificate (along the CA tree from the leaf to the root). All certificates along the "Chain of Trust" should be checked for validity implying requests to each of CAs. The recent launch of Google's Certificate Transparency is aimed to resolve the issue of the unified entry to PKI revocation lists.

As we discussed above historically PGP boasted universal access to the list of revoked certificates, which was realized with the network of PGP key servers. However, delays in synchronization of the PGP keys servers remain one of core challenges affecting efficiency of the certificate revocation and other similar issues. Our experience shows that it takes around one day for the update of the existing PGP certificate among key servers through SKS protocol network as well as spread of the new certificate to majority of PGP key servers.

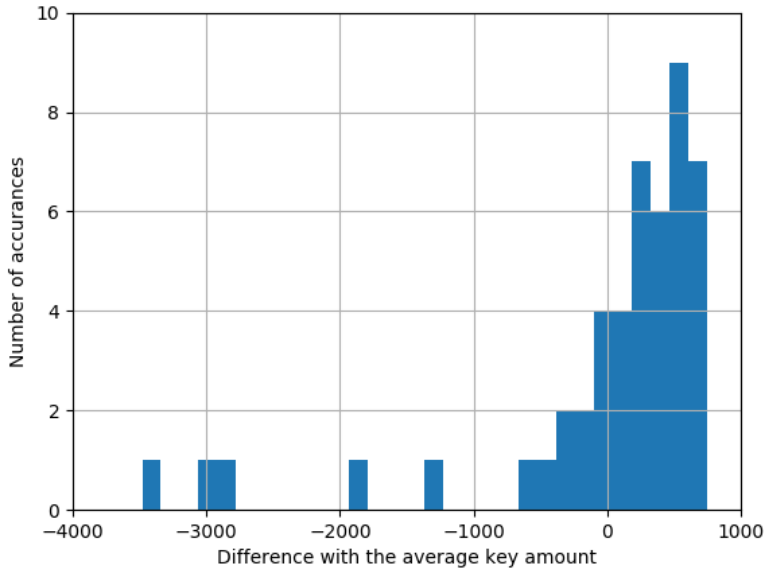

Figure 2: Histogram of difference with the average certificate amount for 48 most active PGP key servers (20.01.2019)

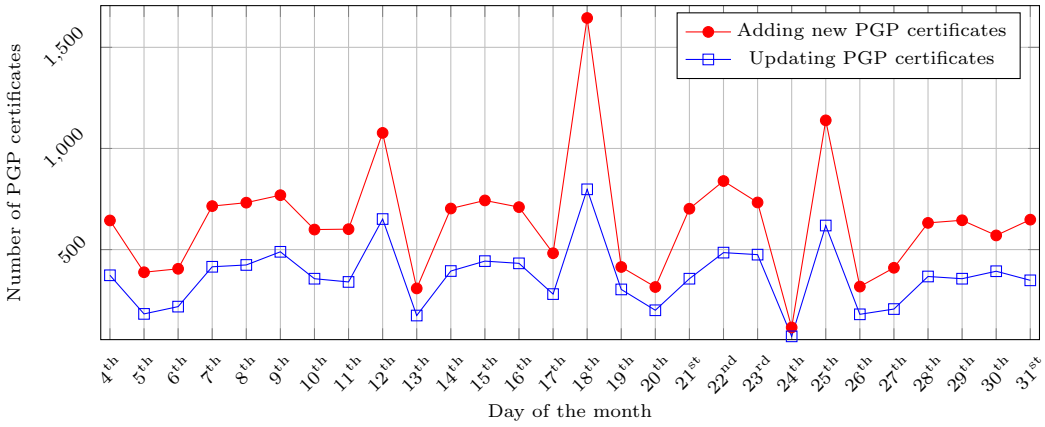

Figure 3: Daily loading of new PGP certificates and updates of the existing certificates for pgp-keys.co.uk server during January 2019

To estimate the synchronization delays, we picked 48 most popular PGP key servers included into SKS protocol pools and scrutinized the number of certificates in their storage. Importantly, the inclusion of the key server to the SKS pools implies its constant connection to the network with full time synchronization based on SKS protocol gossip. It is also important to note that SKS exchange covers the full key storage and is not linked to the loading timestamp. For this reason, it is impossible to remove a certificate from key servers once it was uploaded as it will be reloaded from another key server during synchronization. This means that all PGP key servers will have the same number of certificates in their storage if the loading of new certificates is suspended.

Based on the picked set of PGP key servers we estimated the standard deviation of certificate amount in the key servers' storage at 946 certificates (Figure 2). On the other hand, the average daily load of new certificates (excluding updates of the existing certificates) at pgpkeys.co.uk PGP key server in January 2019 came to 648 certificates (Figure 3). Thus, based on comparison between certificate amount standard deviation and daily loading of new certificates one may come to the conclusion that the delays in PGP key server synchronizations exceed one day. This estimate corresponds to our own experience with the certificate loading to PGP key servers.

The reduction of synchronization latency should diminish risks of using compromised public keys which were already revoked but not yet published with their new status on all key servers. One day of revocation delays can be crucial for victims of private key breaches and in this work we will explore the solutions to reduce the synchronization delays.

3.4 Existing solutions to certificate revocation security challenges

Given the advantages offered by PGP key servers compared to PKI CRLs, the main development in the revocation optimization was concentrated around challenges of PKI revocation lists. Traditional approach to revoked certificate publication used in PKI CRLs lacks efficiency due to delays in publication of updated lists and absence of universal access point to all CRLs. However, recent initiatives like public PKI logs provide some solutions to existing certificate revocation challenges.

As we mentioned above Log-based PKI approach has been proposed as a technique for fast publication of revoked certificates due to the problems stemming from CA breaches and compromised private keys. In Google's Certificate Transparency [18], the most popular log-based PKI, certificates are added to append-only Merkel hash trees of the log servers. Log servers are synchronized with each other for generation of general cryptographic proof showing the existence of the certificate in the log. Unfortunately, despite these benefits, log-based PKIs still have several challenges related to MITM risks [10] and to the deployment process as explained in [20, 21]. Notably, the Merkel tree implementation of the log-based PKI can be naturally realized with blockchain. In addition, blockchain architecture can resolve most of certificate transparency challenges as we will show in this work.

With the introduction of PKI Certificate Transparency the PGP key servers do not seem to remain any more beneficial in terms of revocation efficiency and security. Putting aside PGP key servers' synchronization delays discussed above, we want to draw attention at least to split-world risk affecting both PGP key servers and Certificate Transparency. In our view, PGP key servers may have even greater exposure to split-world risks due to large portion of inactive servers (around 50%). Additionally, Certificate Transparency may benefit from potentially higher level of third-party monitoring conducted by certificate owners and auditors (entities checking identities of certificate owners based on the certificates, the simplest example of auditors are web browsers)

Recently, many studies [19, 8, 14] propose implementation of blockchain technology to build secure PKI systems. Moreover some PKI blockchain-based management frameworks were already deployed and tested on Ethereum testnet [26]. Ethereum smart contracts proved to show good performance for X.509 certificate parsing and validation alongside extensive Chains of Trusts (up to 1200 certificates in a row). Parsing and verification of 1200 certificates took around 8 seconds on ordinary modern notebook.

With the advantages of potential blockchain solutions in PKI compared even to advanced Certificate Transparency, it is important to research the options of blockchain implementations for PGP key servers to resolve at least two of their security challenges, namely, synchronization delays and

split-world risk (or more general MITM risk). We will discuss blockchain-based solution for public key management in the next section.

3.5 Blockchain

Blockchain turns out to be one of the most intriguing technologies in the Internet industry today, mainly due to the success of Ethereum and other smart contract platforms like Hyperledger. The common key characteristics of all blockchain platforms are decentralization, persistency and auditability. Blockchain, also named distributed ledger, is an append-only data structure that stores transactions grouped into blocks to form the hash-chain of blocks. Each block references to the previous (parent) one as it contains the hash of the previous block. The first block of a blockchain is called *Genesis* block which has no parent/preceding block.

Currently most blockchains are used as a ledger for financial or business transactions, however more and more new implementations from different fields start to appear. Applications that require high reliability and full elimination of data manipulation risks can use blockchain. In addition, due to its distributed nature blockchain can avoid the single point of failure situation. In recent years blockchains evolved to allow the execution of arbitrary logic known as *Smart Contracts*. Conceptually the smart contract is an application which runs on the top of blockchain and uses the underlying ordering of transactions to keep consistency of smart contract results between blockchain participants [7]. A smart contract code is executed by a network of *blockchain nodes* that reach consensus on the outcome of the execution, and update the contract's state storage on the blockchain accordingly.

For the Proof-of-Concept development we decided to use a separate instance of Ethereum blockchain which can support any currently available consensus (Proof-of-Work or Proof-of-Authority). A separate instance implies a set of isolated nodes using blockchain only for the purpose of our application. We believe it is better to use a separate blockchain instance for key storage infrastructure due to the following reasons:

- More computation-light consensus algorithms can be used, like Proof-of-Authority instead of Proof-of-Work. This leads to higher potential scalability in terms of transactions per seconds. Proof-of-Authority consensus implies permissioned nature of our blockchain implementation as only selected blockchain nodes (key servers) can confirm blockchain transactions. In this work by permissioned blockchain we will mean the blockchain with predefined signers of transactions.
- Much lower blockchain download time due to limited blockchain size. Basically we will have only PGP certificates in the blockchain.

Notably, in the context of PGP key servers, blockchain provides valuable security features such as write access control based on Ethereum user accounts. Based on our implementation, only administrator and the user corresponding to the Ethereum account recorded in the certificate can update certificate data in the blockchain. Moreover, blockchain-based public key infrastructure, as a public append-only log, naturally provides the benefits of Certificate Transparency with its Merkel trees [8].

3.6 How blockchain naturally solves security challenges of PGP key servers

As we discussed above the core challenges of PGP key servers (also named SKS servers due to their SKS gossiping protocol) include the following:

- **MITM risks** (including split-world risks) stem from potential breach of SSL certificates used by key servers in their web interfaces or from attacker's control of one of inactive key servers. All MITM risks can be avoided due to local replica of blockchain (full-sized repository or just block headers) stored at any client machine and available for reading. Synchronization is done automatically by blockchain client (for instance *geth* for Ethereum).
- **Delays in PGP key servers' synchronization** are crucial for fast and efficient revocations. As we demonstrated above, presently the delays exceed one day and can be reduced to the

latency of 8 block formation coming to 2 minutes by default in Ethereum (can be less). We will explore some mathematical models concerning this subject in the next section.

- Extensive share of **inactive key servers**, reaching half of the total key servers' amount according to SKS protocol scanners[9], implies additional risks of retrieving a revoked certificate as a valid one, etc. With introduction of blockchain, inactive key servers can be easily identified by checking their latest block number. And, more importantly, with blockchain clients like *geth* users do not connect to specified servers (there is no server in distributed system like blockchain). In contrast, by specifying blockchain Id, users assume that blockchain client software will synchronize automatically with other machines holding blockchain with the specified Id and download all recent blocks to their local replica of the blockchain.

As was shown above, in our view all these challenges can be easily resolved with blockchain-based implementation due to its built-in functionalities. Moreover, as we will discuss below, blockchain offers many additional advantages like the system's full history with the ability to reconstruct the state of the storage at any time in the past.

On the other hand, as we discussed the split-world attack and inactive PGP key server attack are based on attacker's control of some PGP key server. Then the question could arise if there is a similar risk in blockchain assuming attacker's control of some node. The answer to this question depends on the consensus used in blockchain and the relative number of nodes the attacker controls. Omitting from consideration Proof-of-Work (PoW) consensus due to its excessive mining costs it was proven that Proof-of-Stake (PoS) consensus has built-in protection from the breach of blockhain's nodes known as Byzantine fault tolerance (for example, for Ethereum PoS refer to [16]). Contrarily, Proof-of-Authority (PoA) consensus with its 50%+1 voting among the authorized nodes from the predefined list may result in some data manipulation risks depending on concrete realization, if one of the authorized nodes is controlled by an attacker. To address this risk the inclusion of new nodes to the list of authorized nodes should be carefully considered. By default such node inclusion is approved by at least 50%+1 of authorized nodes, but this threshold can be increased.

3.7 Mathematical explanation of blockchain's advantages over PGP key servers' SKS protocol

Extensive synchronization delays of PGP key servers can be easily explained using epidemics models considering a certain certificate as an "infection". With the total population size of key servers coming to N we assume that PGP certificates spread among key servers only in some "rounds" of exchange conducted once in predefined time period. We will refer below to this time period as synchronization period.

If $i(t) = I(t)/N$ and $s(t) = 1 - i(t)$ are relative number of infected key servers I and of susceptible key servers $(N - I)$ respectively, we can take into account the following simple differential equations according to so called SI model of mathematical epidemiology (refer, for example, to [15], [17]):

$$\frac{di(t)}{dt} = \beta s(t) i(t)$$

$$\frac{ds(t)}{dt} = -\beta s(t) i(t)$$

Here β is infection rate specifying the portion of susceptible key servers infected by one infected key server during one time unit. With $i_0 = i(0) = 1/N$ the solution of the differential equations above can be easily derived analytically:

$$i(t) = \frac{i_0}{i_0 + (1 - i_0)e^{-\beta t}} \approx \frac{1}{1 + Ne^{-\beta t}} \tag{1}$$

[9] https://sks-keyservers.net/status/

As it can be seen in the formula (1), the dissemination of a certificate among key servers is heavily dependent on the relative number of synchronization peers of a key server, which defines β.

Default configuration of SKS protocol, used for PGP key server synchronization, envisages synchronization period of one hour with its predefined peer key servers specified in the membership list. We assume the synchronization period to remain one hour for majority of key servers. However, the size of membership list is likely to vary significantly from one key server to another. Based on our estimates of the average latency T of a certificate dissemination coming to 36 hours, we can assess the average size of a key server membership list n_m for the SKS pool. Assuming SKS pool size of 50 key servers, we estimate β based on the formula (1) at around 0.2. Then we can write the following formula linking average membership list size n_m and $s(t)$:

$$\frac{n_m}{N} = \frac{\beta \int_0^T s(t)dt}{T}$$

With the presumption of $\int_0^T s(t)dt \approx T/2$ due to certain symmetry of $i(t)$ and $s(t)$ relative to axis $i(t) = s(t) = 0.5$, we estimate $n_m = 5$, i.e. the average size of synchronization membership lists comes to 5 peers for PGP key servers from SKS pool.

To illustrate the implementation of epidemic models to PGP key servers, we simulated dissemination of one certificate among PGP key servers with $\beta = 0.2$ and $N = 50$ (Figure 4). We looked at the model solution with the synchronization period of 1 hours and 10 minutes, which corresponds to default configuration settings of SKS protocol used by PGP key servers and Bitcoin protocol featuring one of the slowest synchronization among blockchain protocols, respectively. Our simulation shows that only the reduction of synchronization period from 1 hour to 10 minutes results in reduction of dissemination latency from 36 hour to 7 hours.

Importantly, the simulation we conducted does not take into account many factors significantly affecting the difference between blockchains' protocols and SKS protocol used by PGP key servers:

- First, in blockchain the synchronization of a given node (i.e. key server) is not limited to pre-defined list of peers (membership list), while SKS protocol provides only membership list synchronization due to security reasons.

- Second, formation of a block in blockchain envisages synchronization of substantial number of nodes depending on consensus. For instance, in case of Proof-of-Authority consensus the block should be approved by more than 50% of all authorized nodes, implying $\beta > 0.5$. For Proof-of-Work consensus a new block is usually fully synchronized among almost all active miners, which also can be considered as authorized nodes.

- Additionally, SI epidemic model better reflects blockchain-based protocols compared to SKS protocol. For instance, "infect and die" model (refer to [12]) can be more suitable for SKS protocol of PGP key servers. Indeed, once the key server communicated using SKS protocol with its peers in the membership list, it does not try to forward the certificate to other key servers not included in the list, and can be considered as "dead" from certificate dissemination view point. Obviously, this results in even slower dissemination of PGP certificates compared to SI model.

- And, finally, given block formation time in Ethereum at 15 seconds by default (could be less), more than 50% of nodes (i.e. key servers) will be synchronized in 15 seconds, while the whole network can be considered as "infected" with a given certificate after around 8 blocks, or around 2 minutes. Waiting for 8 blocks to make sure the transaction was not published in a fork and consequently was accepted by blockchain is a common practice in distributed application development.

With epidemic SI model we can also explain the established practice to confirm the transactions in blockchain with around 8 blocks following the block of a given transactions. This is done to avoid the potential risk of adding the transaction's block to the fork rather than the main blockchain. In

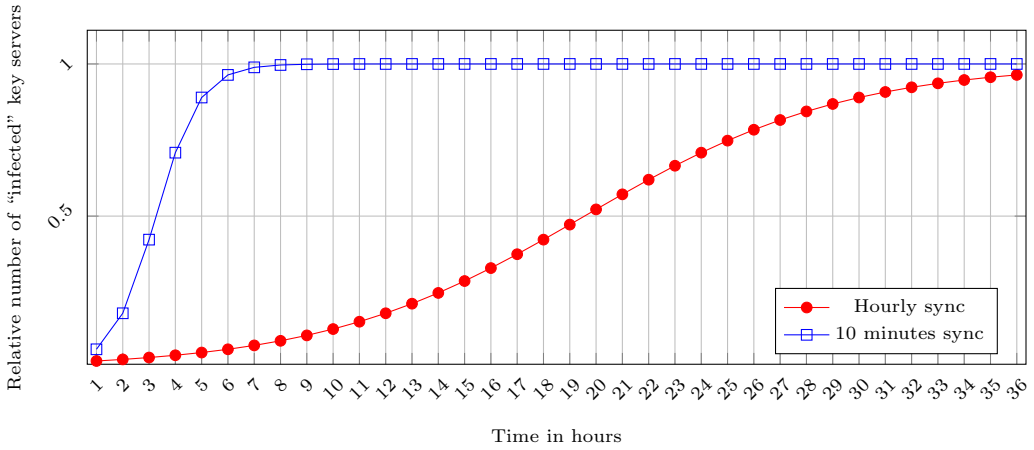

Figure 4: Simulated dissemination of a certificate among key servers according to SI model

formula (1) we assume that at some moment T the whole network is infected with exception of some relatively small portion σ of total number of nodes, or $i(T) = 1 - \sigma$:

$$1 - \sigma = \frac{1}{1 + Ne^{-\beta T}}$$

From this equation we can estimate T:

$$T = \frac{1}{\beta} ln(\frac{(1-\sigma)N}{\sigma}) \approx \frac{1}{\beta} ln(\frac{N}{\sigma})$$

Assuming $\sigma = 0.05$, $N = 50$ and $\beta = 0.8$, we find $T = 8.6$. This means that one needs 8 additional blocks on top of the block with a given transaction to confirm the dissemination of this transaction among blockchain nodes. Our assumption of relatively high β reflects high level of synchronization of the blockchain among authorized nodes (e.g. active miners for Proof-of-Work consensus) In reality Ethereum has a special mechanism of incorporating forks to the existing blockchain, resulting in even higher β.

As a conclusion, our epidemic model shows that in case of PGP key servers implemented based on Ethereum protocol the certificate dissemination latency can be reduced to several minutes from more than one day.

4 Related Work

Blockchain implementations for certificate services or digital identity systems were scrutinized by a number of different prior works. With the exception of Sovrin/Indy embracing as many kinds of authentication as possible on one platform, all these works focused on different aspects of PKI or, like Emercoin, just on safe storage and transfer of certificates eliminating Man-in-the-Middle (MITM) risks.

One of the most interesting ideas in blockchain-based authentication appears to be Sovrin, which in 2017 became known as Hyperledger Indy. Sovrin's concept is based on DID, i.e. distributed ID, the identifier of a certain entity, usually corresponding to a private key [13]. In PGP context such DID can be considered as certificate's fingerprint, effectively SHA-1 hash of public key's core data.

Hyperledger Indy is declared as a public permissioned blockchain and appears to be one of the few public blockchains in Hyperledger ecosystem. The concept of permissioned blockchain stems from Indy's idea of users' digital identity authentication by different entities including authorized

state and public services, etc. The user can forward part of her/his digital identity (for instance driving license only, but not the passport) to some interested third party. Although Indy/Sovrin declares itself as Web of Trust system, in reality it is likely to move towards traditional hierarchical PKI as more state agents are involved. Nevertheless, if users could authenticate some "parts" of each other's digital identity in line with PGP concept of anyone-to-anyone signatures, Indy can be scrutinized as a potential blockchain platform in the framework of this research in the future.

Apart from the overwhelming concept of Indy/Sovrin, there are some concrete implementations of blockchain-based certificate management. For instance, in [26] we deployed PKI management framework on Ethereum's PoW testnet. Golang-based REST service provides a whole range of necessary functionalities, from issuing and revocation of X.509 certificates (including CA certificates), to smart contract-based verification along the whole Chain of Trust up to the root certificate. Surprisingly, smart contract-based parsing and verification of long Chain of Trust consisting of 1200 certificates took around 8 seconds while the verification of this Chain of Trust by Golang code, retrieving the certificates from Ethereum, took 15 sec, or twice slower compared to the smart contracts' performance. In our approach each CA has a separate smart contract holding hashes of all issued certificates, revocation list and the certificate of this CA. To link blockchain infrastructure to certificates, the hybrid X.509 certificates were proposed. Extensions of the hybrid certificates envisaged by X.509v3 standard contain issuing CA's smart contract address and, in case of CA certificate, the smart contract address of this CA.

Regarding blockchain implementation of PKI and Certificate Transparency, an important aspect of research was considered in [25]. Authors looked at the potential techniques to realize TLS handshake based on blockchain's local light node technically built-in to the browser. According to this work, excessive data exchange during TLS handshake can be neglected while storage of block headers for blockchain holding only PKI certificates may come to less than 100 MB.

Implementation of blockchain-based PKI was also announced by Emercoin in its EMCSSH project. Emercoin is a public blockchain quite close to Bitcoin in terms of architecture featuring the hybrid Proof-of-Work and Proof-of-Stake consensus depending on availability of mining capacity. Emercoin does not have smart contracts and stores the certificate hashes into blockchain. This means that the verification of the certificates is not distributed depending exclusively on the code outside blockchain.

Emercoin's EMCSSH is not focusing on Chain of Trust as by default the certificate does not contain links to its parent CA in the extension fields. On the contrary, EMCSSH with just certificate hashes in its blockchain only mitigates the Man-in-the-Middle risk and makes administrators' life easier according to its creators.

Alternatively, Fromknecht et al. [14] leverage Certcoin to implement a blockchain-based PKI, storing domains and their associated public keys. Meanwhile in [8] authors scrutinize the privacy issue of the Certcoin. In [5] the authors propose Blockstack that uses Bitcoin blockchain to provide name registration system where the names are bound with public keys. In this work we mitigate the privacy issue since we will use the standard PGP certificate to develop our framework.

As one can see, by far the majority of research regarding blockchain-based certificate infrastructure was focused on traditional PKI built around SSL certificates, while PGP remained relatively marginal area due to its modest market share in the field of certificate-based identity solutions. On the other hand, as we discussed above in the section 1, the growth in deployment of private blockchains is likely to trigger the demand for decentralized identification based on Web of Trust concept. For instance, Hyperledger Fabric, a leading platform for private blockchains, presently implements user identification based on Fabric CA[10], a traditional centralized PKI based on SSL certificates. In our view the core reason for the present implementation of SSL-based PKI in Hyperledger Fabric is relative management simplicity due to its centralized certificate authority. However, in real-life blockchain deployments the questions may arise regarding the control of this CA by one of the participants.

On the other hand, Hyperledger actively develops Sovrin/Indy based on Web of Trust architecture, but, as we discussed, the inclusion of identification by state authorities can make the system

[10]https://hyperledger-fabric-ca.readthedocs.io

overly centralized, which contradicts the Web of Trust concept.

Given the expected growth of interest to Web of Trust certificate solutions on one hand and Hyperledger Indy's potential centralization on the other, in the long term we do not see alternative to PGP in the area of identification for private blockchains. Although latest version of X.509 standard for SSL certificates envisages multiple signatures by several CAs, PGP certification model with its signatures introduced by any participant looks much more flexible. As we discussed, in PGP different level of trust to a signature can be assigned to different introducing participants. With the expansion of a given private blockchain the trust level to signatures of some blockchain participant can be increased, which affects the trust to certificates this participant signed in the past.

The purpose of our work is to resolve one of the core obstacles in expansion of PGP utilization – its relatively outdated key storage infrastructure compared to new logging solutions in PKI, which deployed by Google and other majors. Additionally, we are hoping to further attract interest of research community to PGP as a leading confirmed solution in Web of Trust certificate-based identification.

5 Blockchain Implementation of PGP Key Server

This section presents the details of our implementations of blockchain-based PGP key servers. Firstly we will explain advantages of a separate instance of Ethereum blockchain rather than using traditional public Ethereum. Then we will justify our approach of incorporating blockchain-related information to PGP certificate. After that, we will discuss the benefits of user rights control based on Ethereum accounts. And, of course, we will describe the main functionality of the smart contract and its two interfaces - command line application and modified Hockeypuck PGP key server.

In the final part of this section we will focus on the performance advantages of PGP key storage deployed on Ethereum blockchain.

5.1 Separate instance of blockchain

Another important consideration regarding blockchain could be the use of Ethereum separate instance (independent set of nodes) rather than traditional public Ethereum blockchain.

The benefits of Ethereum separate instance deployment are the following:

- Flexibility in blockchain consensus, which can substantially increase the performance. We propose to use permissioned blockchain with Proof-of-Authority (PoA) consensus, where transactions are approved by authorized nodes, rather than blockchain with Proof-of-Work (PoW) consensus implying heavy hash calculations conducted by miners. In this context by permissioned blockchain we mean that only authorized nodes can sign and approve transactions.

- Potential difficulties with public Ethereum's blockchain downloading can be easily resolved with deployment of Ethereum separate instance. On January 2019 the size of Ethereum's public blockchain based on "fast sync" option (only recent states of blockchain are kept with full history of blocks and transactions) came to approximately 125GB. Such blockchain size can be explained by massive smart contract deployments. As we are interested only in PGP keys information stored in our smart contracts, it is better to avoid dealing with whole range of data available in public Ethereum. In case of Ethereum separate instance the blockhain will only hold the data related to our application.

- Most importantly, participants of PGP infrastructure will not pay fees for loading data to blockchain. In case of traditional public blockchains, transaction fees are paid to miners.

Present PGP key servers, currently amounting to more than 100 nodes [24], are supported by the community. As PoA, which can be used as a consensus mechanism in PGP key server blockchain, implies no substantial electricity expenses, the community's costs to support key server infrastructure are unlikely to increase with implementation of blockchain technology.

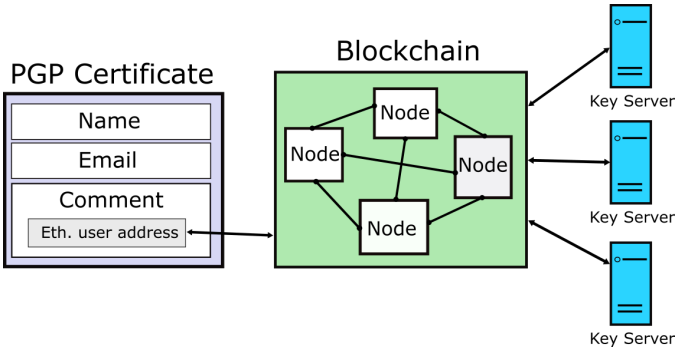

Figure 5: Integrating blockchain user address into PGP certificate

5.2 Design Methodology: incorporating Ethereum user address to PGP certificate

We built our application concept on the same principles implemented in [26] with hybrid X.509 certificates. Once blockchain information (in our case - the address of Ethereum user account) is integrated into a certificate, we can link blockchain access control to user identity and design the whole security concept around it. Instead of expanding certificate data fields using X.509 extensions, we decided to use *comment* field of PGP user identity. Although, to the best of our knowledge, restrictions on use of *comment* field in PGP certificates were not published in the official documentation or in the research papers, many IT security specialists recommend to leave *comment* field blank[11]. Indeed the *comment* field is part of PGP user identity along with name and e-mail address. When a user writes "Work" or "I like strawberries" in the *comment* field of the certificate, the introducer also should confirm by signing this certificate that the certificate holder likes strawberries or will use this certificate at work. On the other hand for our purposes *comment* field is a perfect fit. Introducer should confirm not only the name and email of the certificate holder, but also her/his user address in Ethereum network corresponding to PGP key storage. We use *comment* field in the following format: "blockchain:0x...", i.e. Ethereum user address in hexadecimal format in lower case starting with "0x" is written after keyword "blockchain:"

As we mentioned above the main concept of blockchain-based key server envisages write access restriction assuming that only a user with Ethereum address specified in the certificate's *comment* field and maybe administrator (depending on implementation) can manipulate data related to the certificate. Administrator right control is realized with function modifier *onlyOwner*.

5.3 Access rights based on Ethereum accounts

With Ethereum-based user account control the certificates can be uploaded or modified on a key server only from Ethereum account specified in the certificate. Tracking of all certificate changes with identification of user accounts introducing those changes implies substantial improvements in key server security. Once the attacker was identified, it is possible to scrutinize all his actions and revoke potential threads.

Apart from writing rights of Ethereum accounts specified in PGP certificates, our smart contract design also envisages administrator account which has writing rights for all certificates. We admit that the existence of the admin account implies the single point of failure and, more importantly, contradicts the concept of distributed systems. For instance, to the best of our knowledge, all blockchains do not have built-in admin rights resulting in inability to restore user accounts if the password is forgotten. We kept the admin account in Proof-of-Concept only for demonstration reasons and may remove it in the production version in the future.

[11] https://debian-administration.org/users/dkg/weblog/97

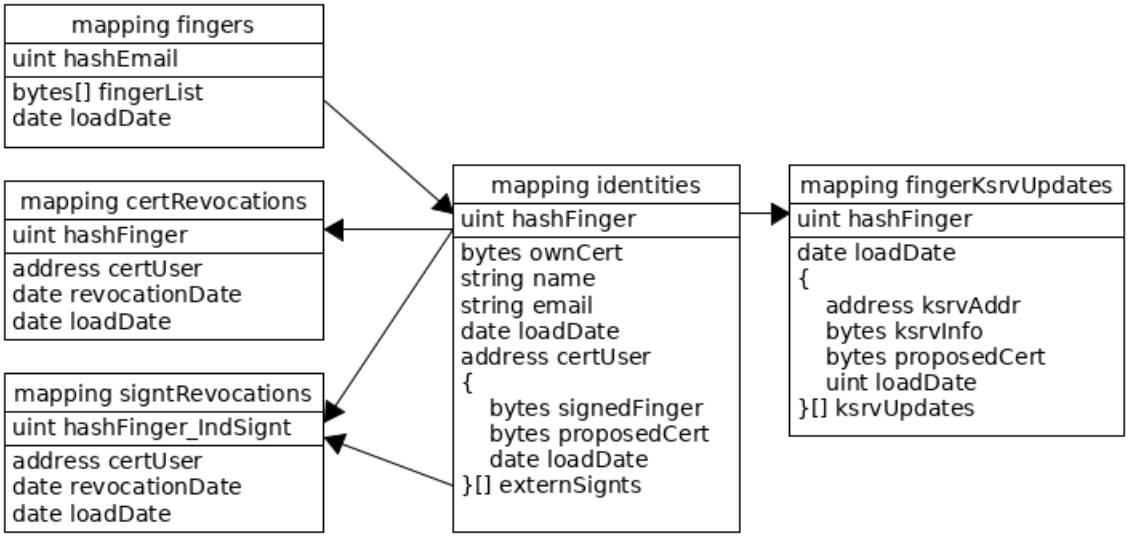

Figure 6: Schema of Smart Contract's data

5.4 Implementation

The proposed Proof-of-Concept of blockchain-based PGP key server includes the following:

- Ethereum smart contract providing key server core functionality.
- Unix command line application *ethpgp* providing simple user interface to the key server's smart contract through IPC protocol of Ethereum's *geth* client.
- Modified version of Hockeypuck SKS server providing synchronization through SKS protocol with public PGP key servers. It might be useful if a user uploads blockchain-linked certificate to public key servers in traditional way (with *gpg* utility or through web interface) rather than directly to Ethereum with *ethpgp* command line application mentioned above.

Notably, Enterprise version of our BlockPGP key server can be deployed without modified Hockeypuck using only Ethereum blockchain and Unix command line application *ethpgp*, if SKS protocol gossiping with other public PGP key servers is not needed.

The BlockPGP software including Unix application, modified Hockeypuck and the smart contract is available as open source at ***https://github.com/alyakubov/blockpgp*** with detailed description of command line options and installation procedure.

5.4.1 Smart contract

In contrast to our PKI blockchain implementation [26] where each CA has its own smart contract we decided to implement a single smart contract to store all PGP certificates' data.

The smart contract provides the following core functions:

- *checkRights*: validates the rights of the user address in the second parameter to change the blockchain data of PGP certificate identified by its fingerprint. Usually user address parameter is the current Ethereum user specified by built-in variable *msg.sender*
- *newCertificate*: uploads PGP certificate to blockchain (mappings *identities* and *fingers*) alongside with all user data including her/his blockchain address (Figure 6). Rights of the user to upload the certificate are verified with *checkRights*. Event *evNewCertificateReturn* is emitted for performance control and error checks.

- *newSignt*: signs (introduces) certificate of another user and uploads the signed certificate to specifically designed storage of proposed certificates, not accessible by other participants (field *externSignts* of mapping *Identities* corresponding to the fingerprint of the introducing certificate). Only the holder of the signed certificate can download the certificate with *getProposedCert* function and/or publish it with *acceptProposedCert* function. Along with event *evNewSigntReturn* used for error control, the event *evProposeCertSignature* is emitted to report the signature for the introducer.

- *revokeCert*: revokes PGP certificate with the user right validation using *checkRights* function (adds data to mapping *certRevocations*). Notably, the user can perform the revocation only with the access to the corresponding Ethereum account without using the revocation key. In our view this may be very convenient as it provides another protected way to revoke compromised certificates. Performance control and error checks are conducted with event *evRevokeCertificateReturn*

- *revokeSignt*: revokes user's signature (introduction) to some PGP certificate of other participant (adds data to mapping *signtRevocations*). In OpenPGP concept it is impossible to revoke signatures, but we decided to include it into our Proof-of-Concept as an experiment. The user right validation is performed with *checkRights* function. Event *evRevokeSigntReturn* is emitted for performance control and error check.

- *acceptPoposedCert*: on certificate holder request copies signed certificate from introducer's proposed certificate storage to signed certificate holder's *ownCert* field of the mapping *identities* (Figure 6). Certificate holder is authenticated with *checkRights* function. Event *evAcceptedCertSignature* is used for performance control and error check.

- *acceptKsrvUpdate*: copies certificate from key servers' proposal storage (mapping *fingerKsrvUpdates*) corresponding to a given fingerprint to certificate holder's *ownCert* field of mapping *identities*. Certificate holder is authenticated with *checkRights* function. Event *evAcceptedKsrvUpdate* is used for performance control and error check.

It is important to note that the ***full history of key server data is available*** thanks to Ethereum built-in functionality. Indeed it is possible to obtain the results of any read-only function (*view* functions based on Solidity's terminology) at any block in the past. The number of the block, when some specific data was changed, can be retrieved based on events we emitted within all core actions of our smart contract.

The proposed implementation of blockchain to PGP key infrastructure implies the following technical challenges:

- ***Smart contract's versioning and code updatability should be handled***, as in contrast to [26] we have a single instance of smart contract's storage and code for all certificate data. There are a number of techniques helping to solve smart contract's code updatability without affecting its data in Ethereum. For instance, we exploited some advantages of Solidity's new opcode *delegatecall* with the placement of all business logic code to solidity libraries. However, presently libraries are statically linked to the smart contract, which holds only shell functions to libraries' calls (linkage to libraries should be conducted before loading the smart contract to blockchain). We will scrutinize techniques of dynamic linking to business logic code in our future work.

- ***Introducer, who signed a certificate, cannot upload it to key server without the certificate holder's acceptance***, which is in line with PGP best practices, but implies additional restrictions on existing PGP key server functionality. Currently an introducer can upload the signed certificate to PGP key server at any time, but preferably he should do it after receiving consent from the certificate holder (according to OpenPGP best practices). Although we believe the certificate update restriction based on user access rights is beneficial for certificate holders, some experts may find significant drawbacks stemming from inability

of updating certificates by a third party without certificate holder consent (we omit admin user from consideration here as administrator is likely to be removed in production version). Importantly, the removal of this access right restriction can be achieved by substantial change of the PoC's architecture and contradicts the concept of the proposed solution.

5.4.2 Unix application

Command line application *ethpgp* was developed with Golang IPC interface to Ethereum *geth* client implying no Man-in-the-Middle (MITM) risks which can be associated with RPC or REST interactions with Ethereum. The application extracts user data from local PGP certificates, parses certificate's blockchain user account from Comment field and connects to Ethereum under this user account. Importantly, Ethereum private key corresponding to this user account should be locally stored in corresponding Ethereum folder of key server blockchain instance. Before each Ethereum connection user is asked to enter a password for Ethereum user.

5.4.3 Modified Hockeypuck key server

Hockeypuck PGP key server was designed with Golang for efficient SKS protocol exchange. It also has a standard web interface for uploading and downloading certificates by end-users. The support for SKS protocol, used for gossiping among key servers for key storage synchronization, is important in situation when an end-user uploades a blockchain-linked PGP certificate using traditional tools like *gpg* command line application or key servers' web interface. In this case certificate will be loaded to keyservers' built-in local storage (for instance, Hockeypuck uses Postgres and MongoDB as storage database, but we used only Postgres version in our research). The Modified Hockeypuck should be able to load PGP certificates with blockchain specification from built-in local storage to Ethereum and vice versa.

To arrange an interaction between blockchain-based distributed storage for PGP certificates and built-in local storage of public PGP key servers we modified the *storage* package of Hockeypuck's Postgres support to load certificates to both Postgres and Ethereum.

Technically the Ethereum functionality of Hockeypuck PGP key server is implemented in the following way:

- When an end-user uploads PGP certificate to key server built-in storage (for example local database like Postgres) using traditional tools the *Insert* or *Update* function of the *storage* package is called. We modified these functions providing the upload to Ethereum as well if the certificate was never loaded to Ethereum before.

- When an end-user uploads PGP certificate directly to Ethereum using *ethpgp* tool we upload it from Ethereum to key server built-in local storage using Ethereum event listener. If the certificate (its current version with given signatures, etc.) was ever uploaded to Ethereum it will not be uploaded again.

To avoid duplication, it is important to check if the certificate was ever uploaded from a key server's built-in local storage to Ethereum or vice versa. We use MD5 hash comparison of the certificate content, as the same hash check is implied in SKS protocol for duplication control.

5.5 Performance

As we discussed above performance is one of the primary reasons for introducing blockchain to PGP key infrastructure. The synchronization latency of Ethereum nodes (PGP key servers in our case) is defined by time period of block formation and can be configured at the launch of blockchain in genesis block. By default the block formation period in Ethereum is 15 seconds.

There are on average around 1000 writing transactions in PGP key servers per day (adding new certificates and updates of the existing ones) which corresponds to daily certificate loading of pgpkeys.co.uk server in January 2019 (we explored it in the section 3.3 dedicated to assessment of PGP key server synchronization delays). Additionally the estimate of daily PGP certificate loading

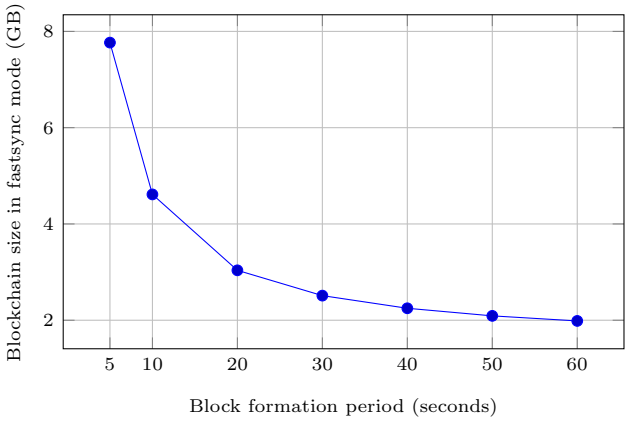

Figure 7: The annual growth of BlockPGP blockchain fastsync size (GB) vs block formation period (seconds)

to key servers at 1000 is confirmed by [23]. According to Ethereum developers this implies no difficulties with scalability in the future as Ethereum can perform more than 10 transactions per second under PoW consensus and much more under PoS/PoA. To check this, we conducted a test for PoA consensus with block formation period of 1 second set as a parameter in genesis block. We used four blockchain nodes running Red Hat Enterprise Linux and we confirmed the load of the data to blockchain with Ethereum events, thus making sure that no data is lost. During the test Ethereum performance reached 10 transactions per second, implying extensive scalability from the present level of 1 transaction in more than 1 minute on average.

However, with the reduction of block formation period the blockchain size increases substantially and becomes important factor of performance. For example, the size of public Ethereum blockchain in January 2019 came to impressive 125 GB for nodes under fastsync mode and manyfold more than that under full mode. It is for avoiding problems with large-sized blockchain synchronization we decided to use a separate instance of Ethereum containing only data corresponding to our project (basically only data relating to PGP certificates and single instance of smart contract with libraries).

To estimate the annual size increase of our separate instance of blockchain (fastsync node) we used the size of empty block coming to 1 KB and the size of certificate add/update transaction amounting to 4 KB. We assume the daily amount of certificate transactions (both updates of existing certificates and additions of new ones) will come to 1000. Based on these assumptions, we estimate an annual increase in blockchain size of around 2 GB if empty block generation is modest (block formation period is 60 seconds). With block formation period of 5 seconds the blockchain size growth in one year may increase almost fourfold and reach 8 GB due to sizable amount of empty blocks (Figure 7). Thus the block formation period should balance number of transactions per seconds and the size of blockchain. On the other hand, the issue of blockchain's extensive size can be solved with suspension of empty block generation.

Reading performance from the local replica of blockchain remains relatively fast as it was shown in our previous work dedicated to PKI framework on Ethereum [26]. For instance, we demonstrated that parsing of X.509 PKI certificates inside smart contract (8 seconds) is almost twice faster than that with standard Golang libraries (15 seconds) for trust paths of 1200 certificates.

Another issue regarding performance might be the dumping of the existing key database to a fresh new PGP key server, which takes several hours for Hockeypuck's Postgres version. The benefit of blockchain in this area stems from automatic loading of the existing blocks to the local replica, which is done virtually at the speed of data transfer connection as no SQL transaction settlement is required for each certificate in contrast to Postgres-based key server dump.

It is important to point out that to estimate blockchain size we used fastsync node (default *geth* command option). It implies automatic pruning (removal) of state data history from blockchain, while transaction history will be kept starting from genesis block.

In addition to fastsync nodes and full nodes Ethereum team recently launched an experimental implementation of the light mode for deployment of fast performing nodes on mobile devices[12]. The light Ethereum node stores locally only block headers starting from certain rather recent block while the rest of the data is provided by the full Ethereum nodes found by the light node with discovery protocol. We will focus on mobile device deployment of our *ethpgp* command line application in our future work.

6 Conclusions and Future Work

Key servers are an important part of OpenPGP's Web of Trust infrastructure. Blockchain-based Proof-of-Concept of PGP key server proposed in this work solves a number of PGP key infrastructure's challenges and improves its performance. First, the developed Proof-of-Concept provides the right to upload certificates to key servers only to the certificate holder, which reduces the risk of downloading compromised certificate from the key server. Second, blockchain resolves Man-in-the-Middle risk for key servers (including split-world risks). Third, blockchain accelerates the synchronization among key servers to several minutes from around one day. Moreover, blockchain provides full history of a key server's states based on its built-in functionality. As a part of our Proof-of-Concept we modified an existing Hockeypuck SKS server to arrange integration of blockchain-based solution to existing public key server infrastructure. Importantly, to achieve user right control we incorporated blockchain-related data to PGP certificates.

Notably, if we use a separate instance of blockhain (independent nodes dedicated from blockchain view point exclusively to key server tasks), the calculation-light consensus mechanisms can be used. This means that the total costs associated with the PGP infrastructure support by the community are unlikely to grow.

In the future, we plan to develop smart contract-based parsing and verification of PGP certificates. Additionally, more attentions should be drawn to updatability of smart contracts' code. And, the last but not the least, with the advances of Ethereum light client introduction we plan to extend this work toward a mobile version of BlockPGP software.

References

[1] The GNU OpenPGP privacy handbook, 1999. Available at `www.gnupg.org/gph/en/manual/book1.html`.

[2] Black tulip: Report of the investigation into the diginotar certificate authority breach, 2012. Available at `http://arkanoid.hu/stuff/Black%20Tulip%20-%20Diginotar%20Report.pdf`, retrieved on 1.05.2019.

[3] Types of SSL certificates – choose the right one, 2014. Available at `https://www.symantec.com/connect/blogs/types-ssl-certificates-choose-right-one`.

[4] Carlisle Adams and Steve Lloyd. *Understanding PKI: concepts, standards, and deployment considerations*. Addison-Wesley Professional, 2003.

[5] Muneeb Ali, Jude C Nelson, Ryan Shea, and Michael J Freedman. Blockstack: A global naming and storage system secured by blockchains. In *USENIX Annual Technical Conference*, pages 181–194, 2016.

[6] Hiroaki Anada, Junpei Kawamoto, Jian Weng, and Kouichi Sakurai. Identity-embedding method for decentralized public-key infrastructure. In *International Conference on Trusted Systems*, pages 1–14. Springer, 2014.

[7] Elli Androulaki, Christian Cachin, Angelo De Caro, Alessandro Sorniotti, and Marko Vukolic. Permissioned blockchains and hyperledger fabric. *ERCIM News*, 2017(110), 2017.

[12] https://github.com/ethereum/wiki/wiki/Light-client-protocol

[8] Louise Axon and Michael Goldsmith. PB-PKI: A privacy-aware blockchain-based PKI. In *Proceedings of the 14th International Joint Conference on e-Business and Telecommunications (ICETE 2017) - Volume 4: SECRYPT, Madrid, Spain, July 24-26, 2017.*, pages 311–318, 2017.

[9] Germano Caronni. Walking the web of trust. In *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2000.(WET ICE 2000). Proeedings. IEEE 9th International Workshops on*, pages 153–158. IEEE, 2000.

[10] Laurent Chuat, Pawel Szalachowsky, Adrian Perrig, Ben Laurie, and Eran Messeri. Efficient gossip protocols for verifying the consistency of certificate logs. In *2015 IEEE Conference on Communications and Network Security (CNS)*, 2015.

[11] Dave Cooper, Stefan Santesson, Stethen Farrel, Sharon Boeyen, Rassel Housley, and Timothy Polk. Internet X.509 public key infrastructure certificate and certificate revocation list profile. RFC 5280, 2008. Available at `https://www.ietf.org/rfc/rfc5280.txt`.

[12] Patrick T Eugster, Rachid Guerraoui, A-M Kermarrec, and Laurent Massoulié. Epidemic information dissemination in distributed systems. *Computer*, 37(5):60–67, 2004.

[13] Sovrin Foundation. Sovrin: A protocol and token for self-sovereign identity and decentralized trust. A White Paper, 2018. Available at `https://sovrin.org/wp-content/uploads/Sovrin-Protocol-and-Token-White-Paper.pdf`.

[14] Conner Fromknecht, Dragos Velicanu, and Sophia Yakoubov. Certcoin: A namecoin based decentralized authentication system. *Massachusetts Inst. Technol., Cambridge, MA, USA, Tech. Rep*, 6, 2014.

[15] Herbert W. Hethcote. The mathematics of infectious diseases. *SIAM Review*, 42:599–653, 2000.

[16] Akshita Jain, Sherif Arora, Yashashwita Shukla, et al. Proof of Stake with Casper the friendly finality gadget protocol for fair validation consensus in Ethereum. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 3, 2018.

[17] Matt.J. Keeling and Ken.T.D. Eames. Networks and epidemics models. *Interface Journal of the Royal Society*, 2:295–307, 2005.

[18] Ben Laurie, Adam Langley, and Emilia Kasper. Certificate transparency. RFC 6962, 2013.

[19] Karen Lewison and Francisco Corella. Backing rich credentials with a blockchain PKI. *Tech. Rep.*, 2016.

[20] Stephanos Matsumoto and Raphael M Reischuk. IKP: Turning a PKI around with blockchains. *IACR Cryptology ePrint Archive*, 2016:1018, 2016.

[21] Stephanos Matsumoto, Pawel Szalachowski, and Adrian Perrig. Deployment challenges in log-based PKI enhancements. In *Proceedings of the Eighth European Workshop on System Security*, page 1. ACM, 2015.

[22] Damian Poddebniak, Christian Dresen, Jens Müller, Fabian Ising, Sebastian Schinzel, Simon Friedberger, Juraj Somorovsky, and Jörg Schwenk. Efail: Breaking S/MIME and OpenPGP email encryption using exfiltration channels, 2018.

[23] Alexander Rucker. An efficient PGP keyserver without prior context. 2017. Available at `http://www.scs.stanford.edu/17au-cs244b/labs/projects/rucker.pdf`, retrieved on 1.05.2019.

[24] Alexander Ulrich, Ralph Holz, Peter Hauck, and Georg Carle. Investigating the OpenPGP Web of Trust. In *ESORICS 2011: Computer Security*, pages 489–507, 2011.

[25] Ze Wang, Jingqiang Lin, Quanwei Cai, Qiongxiao Wang, Jiwu Jing, and Daren Zha. Blockchain-based certificate transparency and revocation transparency. In *Financial Cryptography and Data Security, Springer International Publishing*, 2018.

[26] Alexander Yakubov, Wazen Shbair, Anders Wallbom, David Sandra, and Radu State. A blockchain-based PKI management framework. In *The First IEEE/IFIP International Workshop on Managing and Managed by Blockchain (Man2Block) colocated with IEEE/IFIP NOMS 2018, Tapei, Tawain 23-27 April 2018*, 2018.

[27] Jiangshan Yu and Mark Ryan. Evaluating web PKIs. In *Software Architecture for Big Data and the Cloud*, pages 105–126. Elsevier, 2017.