A Blockchain-Based PKI Management Framework

Alexander Yakubov *, Wazen M. Shbair *, Anders Wallbom †, David Sanda †, Radu State *
* University of Luxembourg, SnT, 29, Avenue J.F Kennedy, L-1855 Luxembourg
Email:{alexander.yakubov, wazen.shbair, radu.state}@uni.lu
† Nexus Group
Email:{anders.wallbom, david.sanda}@nexusgroup.com

Abstract—**Public-Key Infrastructure (PKI) is the cornerstone technology that facilitates secure information exchange over the Internet. However, PKI is exposed to risks due to potential failures of Certificate Authorities (CAs) that may be used to issue unauthorized certificates for end-users. Many recent breaches show that if a CA is compromised, the security of the corresponding end-users will be in risk. As an emerging solution, Blockchain technology potentially resolves the problems of traditional PKI systems - in particular, elimination of single point-of-failure and rapid reaction to CAs shortcomings. Blockchain has the ability to store and manage digital certificates within a public and immutable ledger, resulting in a fully traceable history log. In this paper we designed and developed a blockchain-based PKI management framework for issuing, validating and revoking X.509 certificates. Evaluation and experimental results confirm that the proposed framework provides more reliable and robust PKI systems with modest maintenance costs.**

I. Introduction

Public Key Infrastructure (PKI) provides a secure mean of authenticating identities over Internet. It defines the policies and procedures needed to issue, manage, validate and distribute digital certificates in order to use public-key encryption securely [1]. PKI management of public-keys is usually based on the certificate standard X.509 which provides verification of ownership of a private-key by some external entity (certificate authority). The X.509 certificate is defined as a data structure that binds public-key values to subjects (e.g., domain names). The binding is asserted by trusted Certificate Authorities (CA) digitally signing each certificate. The CA may base this assertion by profoundly validating the identify of the private certificate holder [2].

Recently, flaws in the security procedures of well-known CAs show that the security of CAs is subject to operational errors. As CAs appear to be single points of failure in PKI, CA's errors or breaches have resulted in unauthorized certificates being issued [3]. For instance, one of a high-profile event 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 for high-profile domains, including one for Google.com. As a result web browser vendors were forced to blacklist all certificates issued by the Dutch CA DigiNotar after more than 500 fake certificates were discovered [4].

In another case, DigiCert Sdn. Bhd., a Malaysian subordinate certificate authority, 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.) [4]. There were also certificate breach problems with TrustWave, a large U.S.-based certificate authority. TrustWave admitted that it issued subordinate root certificates to one of its customers so the customer could monitor traffic on their internal network. Subordinate root certificates allow their holders to create SSL certificates for nearly any domain on the Internet. Although TrustWave has revoked the certificate and stated that it will no longer issue subordinate root certificates to customers, it illustrates just how easy it is for CAs to make missteps and how severe the consequences of those missteps might be.

A. PKI issues and existing solutions

Two approaches were proposed to solve SSL/TLS PKI issues: Log-based PKIs schema, and decentralized networks of peer-to-peer certification, known as Web of Trust (WoT).

Log-based PKIs approach has been proposed as an emerging solution to the problem of misbehaving CAs. 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. Hence any misbehavior by CAs will be detected by users and servers. Google's certificate Transparency (CT) [5] is the most widely deployed log-based PKI, and it is currently available in both Chrome and Firefox. In parallel, many proposals intend to extend the features of Log-based PKI schema by support for revocation and error handling. Unfortunately, despite these benefits, log-based PKIs still have several challenges related for instance to certificate revocation as explained in [6], [7].

Web of Trust (WoT) approach is entirely decentralized: users are able to designate others as trustworthy by signing their public key certificates. By doing so, a user accumulates a certificate containing his public key and digital signatures from entities that have deemed him trustworthy. The certificate is then trusted by a third party if it is possible to verify that the certificate contains the signature of someone she/he trusts. This system benefits from its distributed nature because it removes any central point of failure. However, it makes it difficult for new or remote users to join the network, since some existing member of WoT typically must meet with the new user in person to have her/his identity verified and public key signed for the first time. Also, unlike a CA-based approach, the WoT has no way to deal with key revocation. A user is limited to choosing another user to be her/his ”designated revoker” with the rights to revoke her/his certificate when the private key is lost or compromised. The current practical approaches depend on periodically pushing revocation lists to browsers. However, such method makes invalid certificates to be trusted until the browser’s revocation lists get the the next push [1].

Blockchain-based PKI inspired many studies [8], [9], [10] proposing blockchain technology to build secure PKI systems. Their main argument is that blockchain-based solution can merge the benefits of the Log-based PKIs and the WoT approaches, and solve some of the problems with conventional PKI system. On one hand blockchain resolves the potential points-of-failure of log-based PKI approach and deployment issues as we will discuss later. On the other hand the blockchain-based approach mitigates the WoT needs for new certificate holders to prove the trustworthiness by existing network members.

Our main contribution is a complete blockchain-based PKI framework to manage X.509 certificates. Our framework includes several innovations. First, we extend the standard X.509 certificate to be compatible with blockchain-based PKI approach, thanks to X.509 extension fields that we used to embed blockchain meta data. Second, we design and implement a blockchain-based PKI which provides reliable management for digital certificates.

The rest of the paper is organized as follows: Section II provides a background of the certificate validation mechanism, named chain-of-trust and an introduction to the Blockchain technology and how it helps to build secure PKI systems. Section III explores related work on using blockchain to implement PKI. In Section IV we present our blockchain-based PKI solution. Evaluation results and discussion are provided in Section V. Finally, Section VI concludes the paper.

II. BACKGROUND

In this section, first we present the idea behind the chain of trust, which used to validate X.509 certificates. Then we provide a brief introduction to the blockchain technology and how it can be used to build PKI management framework.

A. Chain of trust

The classical PKI systems are CA-based. CAs issue a signed certificate, specifically complying with the X.509 standard, that certifies an entity’s ownership of a public key. For instance, when a user logs into Twitter through a web browser, first the web browser validates the claimed certificate which holds Twitter’s public key by checking the CA of the given certificate. Usually web browsers are pre-configured to accept certificates from certain known CAs. Therefore, in order for a certificate to be trusted, it must has been issued by a Root-CA that exists in the trusted store of the user browser or device, or by sub-CA that has been trusted with Root-CA signature. Typically, Mozilla products come with 154 root certificates [11]. Also, Apple, Microsoft and Google have their own store of trusted root certificates embedded in their products.

The link between a given certificate and the Root certificate is known as a *Chain of Trust*. Importantly, chain of trust may include any number of sub-CA certificate between a given certificate and the Root-CA certificate. However, the X509v3 has an extension named *Basic Constraints* and this extension can limit the maximum depth of valid certificate chain (chain of trust) [2].

Figure 1 illustrates a certification path from an end-entity certificate to the Root-CA, where the chain of trust begins. Hence, if the end-entity certificate was not issued by a trusted CA, the web browser will then check to see if the certificate of the issuing CA was issued by a trusted CA, and so on until either a trusted CA is found, or no trusted CA can be found in the chain then the browser will usually display an error.

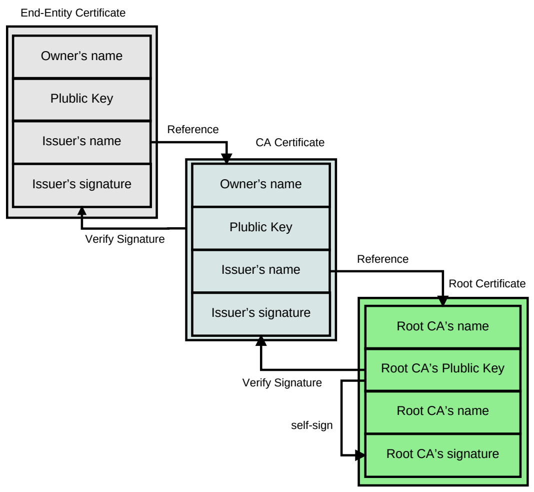

Fig. 1: Chain of trust

B. Blockchain Technology

Blockchain or distributed ledger turns out to be the most intriguing technology in the Internet industry, mainly after witnessing success of Bitcoin. Currently, most blockchain platforms are used in the financial applications, however more and more new applications for different fields start to appear. Applications that require high reliability and full elimination of data manipulation risks can use blockchain. In addition, blockchain is distributed, so it avoids the single point-of-failure situations.

In the recent years blockchains have been 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 execution results between peers [12]. For example, Ethereum blockchain supports complex and stateful Turing-complete language, like Solidity[1], that can be used to program and define wide range of application scenarios [13].

In the context of PKI, the blockchain technology provides valuable security features such as certificate revocation,

[1]http://solidity.readthedocs.io/en/develop/

elimination of central points-of-failure and a reliable transaction record. For instance with the fast certificate revocation, blockchain-based PKIs can instantaneously isolate the infected CA and the corresponding certificates without the need to wait until the next update of Certificate Revocation Lists (CRLs). Also, blockchain-based PKI, as a public append-only log, naturally provides the certificate transparency (CT) property proposed by Google [9].

In this work, we selected the Ethereum platform and Solidity smart contract programming language, since they have large open-source community making the development process much more efficient.

III. Related work

Blockchain implementations to build PKI system was scrutinized by many different prior work. In [14] the authors propose Blockstack that uses Bitcoin blockchain to provide name registration system where the names are bound with public keys.

Similar to Blockstark blockchain-based PKI is realized in Emercoin[2] (EmerSSH 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. EmerSSH 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 EmerSSH stores certificate hash to blockchain thus according to authors mitigating the man-in-the-middle risk. Once the hash of certificate is loaded into blockchain the secure connection using the certificate's public key is established and the exchange of brand-new keys is conducted for each connection.

Fromknecht et al. [10] leverage Certcoin to implement a blockchain-based PKI, storing domains and their associated public keys. Meanwhile in [9] authors scrutinize the privacy issue of the Certcoin.

All the aforementioned studies propose pure blockchain-based approaches. However, in this work, we propose to use the common standard X.509v3 certificate with a minor addition to the extension fields with blockchain related information [2]. Thus, the exended X.509 certificate it can be validated by the classical CA-based chain of trust or using blockchain-based PKI framework. To the best of our knowledge, we are the first to propose such a hybrid certificate. As future work, we plan to develop a plug-in for web browsers that validates claimed certificates using public blockchain.

IV. Blockchain-Based PKI Framework

This section presents our framework for managing PKI in a blockchain platform. Firstly we will provide an overview of the main benefits of the proposed approach. Then, we shall explain our methodology for arranging the main players in the PKI system (i.e., Root CA, Sub CAs, Certificate holders).

A. Overview

Our blockchain-based PKI framework supports the revocation of the certificate, which is a real issue in the traditional PKI systems. Moreover, as it is impossible to delete information from blockchain, only a parent CA can mark a certificate issued by him as revoked. Thus, any misbehavior of a CA regarding certificate revocation will be also traced and noticed by all other entities.

B. Design Methodology

The design of blockchain-based PKI is based on hybrid X.509 certificates, as shown in Figure 2. A certificate contains certain information on PKI environment in the extension fields. The value of extension fields are as follow:

- ***Subject Key identifier***: holds the certificate's owner identity.
- ***Blockchain name***: holds the name of the blockchain platform. Currently, we use Ethereum public blockchain but we envision to cover more platforms.
- ***CA key identifier***: holds the smart contract address of the current CA, if this is CA certificate. For non-CA certificates this field is empty.
- ***Issuer CA identifier***: holds the address of the smart contract of the CA which issued this certificate. It allows validator to find parent CA smart contract in the blockchain and check if the certificate with the corresponding hash was issued and was not revoked. For root certificates this field is empty.
- ***Hashing algorithm***: holds information regarding hashing algorithm which used in calculation of the certificate's hash loaded into blockchain.

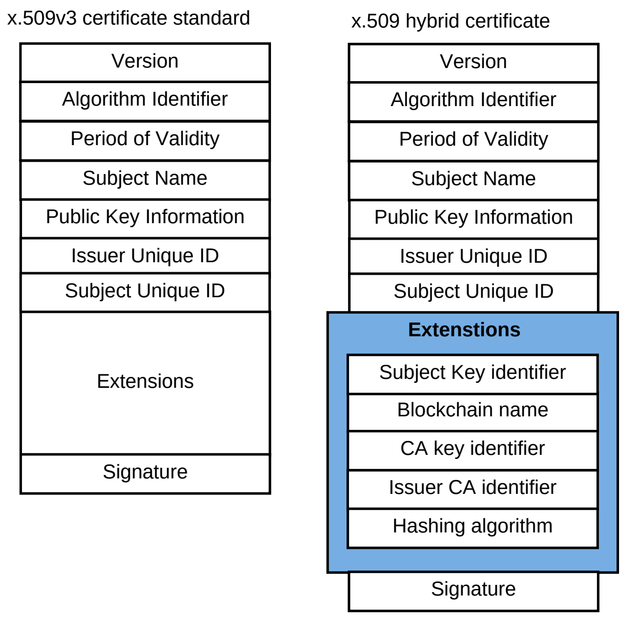

Fig. 2: Standard vs. Hybrid X.509 certificate

[2]https://emercoin.com/en/tech-solutions/

Based on the extension structure above our framework envisages three types of certificates: Root-CA, Sub-CA and End-user certificates. Table I illustrates the hierarchy of blockchain hybrid certificates. The first row presents a Root CA where the certificate is issued and signed by himself, so there is no issuer CA as shown as a Issuer CA ID in the fifth column (i.e 0x00000000). The Sub-CA's certificate is given in the second row - it has been issued by a Root-CA and the Issuer-CA ID points to the Root-CA. The last row holds end-user certificate issued by Sub-CA. In the next section we explain how we realized this structure on a blockchain platform.

TABLE I: Blockchain hybrid certificates hierarchy

Cert.	Issued By	Issued To	CA Contract ID	Issuer CA ID
RootCA	RootCA	RootCA	0x1234xxxx	0x00000000
SubCA	RootCA	SubCA	0x5631xxxx	0x1234xxxx
EndUser	SubCA	End user	-	0x5631xxxx

C. Architecture

The main idea of our framework is that each CA has a dedicated smart contract that includes the following information:

- Array with the hashes of the issued certificates. It also may include expiration date for each certificate and some other technical information.
- Mapping of the revocation data referenced by the certificate hash. If a certificate is revoked the CA which issued this certificate adds the revoked certificate's data to this mapping.

Moreover, if a certificate is a CA's certificate the certificate itself is also loaded into the corresponding CA's smart contract. As certificate contains address of the parent CA smart contracts, it allows to validate the whole certificate authority tree (chain of trust) from the leaf up to root certificate.

D. Implementation

The proposed Blockchain-based PKI framework includes three main parts: Restful service, certificate validation and the Web user interface basically used for testing.

1) Restful service: it has been developed with Golang as a stand alone web server providing access to Ethereum public blockchain. It provides all functionality of issuing, revocation and validation of the certificates. Importantly, the validation is conducted "for free" from the view point of public blockchain's crypto-currency costs, as validation does not add or alter data in the blockchain.

The Restful service provides the following core functions:

- *Enroll-user*: adding a hash of certificate to a given CA's smart contract. As an alternative to providing hash as a parameter, the certificate can be uploaded to the Restful service. In this case the hash is calculated based on the uploaded certificate.
- *Blacklist-user*: revokes a certificate, i.e. moves the certificate (either ordinary or CA) from the white-list to the black-list. Technically this is achieved by adding the certificate hash reference to revocation mapping in the smart contract.
- *Create-contract*: creates empty contract for a new CA. This is called by the parent CA when issuing the certificate for its sub-CA.
- *Populate-contract*: after creating empty smart contract for a sub-CA the parent CA has to upload into it the sub-CA's certificate containing the parent smart contract address and the sub-CA's smart contract address in the extension fields. After the population of sub-CA smart contract with the certificate by its parent CA the sub-CA's Ethereum account address is recorded into *owner* variable of the smart contract thus providing the write access only to the sub-CA.
- *Validate-cert* (view/constant function): validation of the certificate from leaf to the root of the CA tree. Importantly, verification of the certificates can be conducted by Restful service based on Golang code or based on the call to the separate validating smart contract. As we discussed above, all validation is conducted at zero cost.

The first four functions of the Restful service functionality imply the authorization of an Ethereum user corresponding to the parent certificate authority (CA). The important feature about Restful service functionality is intentionally multistage initiation of sub-CA enrollment (adding sub-CA certificate to the list of approved certificates in the smart contract of the parent CA). In contrast to traditional end-user certificate which is initiated just with the Restful service's function *Enroll-user*, sub-CA certificate is initiated with the following steps:

- Parent CA should create empty smart contract for sub-CA with the function *Create-contract*. Now parent CA can generate the hybrid certificate putting the new smart contract address into the corresponding field of the certificate extensions.
- Then parent CA populates sub-CA's new smart contract with the generated certificate using *Populate-contract* function of the Restful service. After execution of the *Populate-contract* function the writing rights to the new smart contract are transfered exclusively to the sub-CA with the filling of the sub-CA Ethereum address into the new smart contract's *owner* field
- And, at last, the hash of the smart contract is recorded to the parent CA's white list with *Enroll-user* function

2) Validation: The validation module contains a smart contract that allows to validate the chain of trust for a given certificate (path from the leaf, or end-entity certificate, to the root in the CA tree). Importantly, the smart contract valiadtion is independent from the Restful service's Golang code valiadtion, an alterntive approach to certificate validation also implemented in our framework. Obviously, both of these two validation approaches imply no payments of cryptocurrency, as blockchain is not altered.

3) User Web interface: The user interface allows clients to test the whole application - to add CA and end-user certificates at different level of CA tree under different CA accounts, revoke certificates, etc. Obviously, web interface can be considered as a shell to the Restful service mentioned above. It is worth mentioning that the test web interface has its own smart contract that stores some data including the links from a parent CA to smart contracts of its sub-CAs. This allows to navigate from root to the leafs (end-entity certificates) of the certificate tree, obviously under conditions that the given chain of trust was loaded with the Web interface.

E. Advantages of Blockchain-based PKI

A blockchain-based PKI has the following advantages over a traditional PKI. First, the validation of a certificate and its CA certificate chain is simple and fast. Second, blockchain-based PKI solves a longstanding problem of traditional PKIs by not requiring the use of a service that issues certificate revocation lists (CRLs) thanks to blockchain synchronization between network's nodes where any modification to the state of a certificate will be instantaneously notified to the all nodes [8].

Another important aspect in the context of Internet security is that the blockchain-based PKI provides flexible protection against the man-in-the-middle (MITM) attacks. Traditionally, MITM is considered as a major security risk implying attacker to hijack a browser's connection for a given websites by presenting a valid certificate (i.e., forged public-key) for that domain. For users and web browsers it is difficult to identify the replacement of certificate in case the related CA has been hacked by the attacker [15]. Blockchain-based PKI approach makes MITM attacks virtually impossible as when a CA publishes or revokes the public key of a website/domain on the blockchain, the information will be distributed across thousand nodes, so tampering the public-key will be (theoretically) out of the question [16]. Traditional PKI resolves MITM risks by embedding Root CA certificates into browser installations, thus artificially expanding CA entrance barriers and increasing the time necessary for Root CA certificate revocation.

V. Evaluation and Experimental Results

This section presents different evaluations tests. We start with general performance evaluation, then we measure the cost of blockchain-based functions and we finish with the limitation of the blockchain-based PKI platform.

A. Performance

To prove the efficiency of the PKI based on Ethereum's smart contracts, we conducted a number of experiments on public Ethereum Testnet (Rinkeby)[3] regarding the performance of CA certificate verification through the full path from the leaf (a given CA certificate) to the root (chain of trust). The idea of the experiment is the performance comparison between the smart contract-based verification and the Golang code verification of the Restful service.

Restful service verification : initially our certificate validation though full chain of trust was based on the Restful service that gets the certificate for each CA from the blockchain, parses the certificate with Golang libraries to extract the parent CA's smart contract address and then checks the certificate validity based on the corresponding hash stored in the parent CA's smart contract.

[3]https://www.rinkeby.io

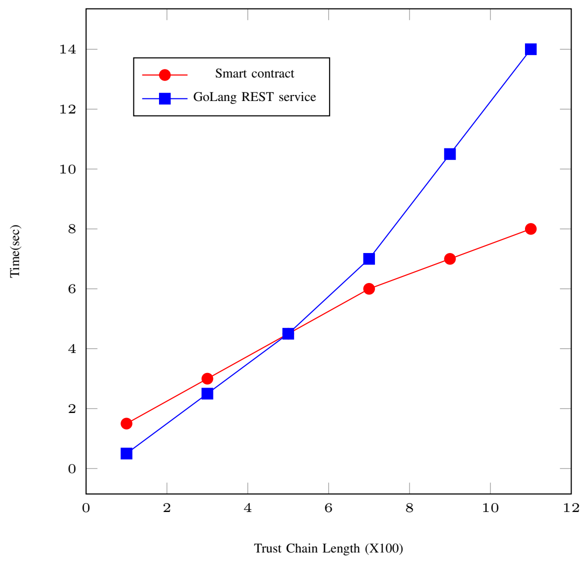

Fig. 3: CA certificate trust chain verification

Smart contract-based verification: an alternative approach that proves to be significantly more efficient. The idea is that a dedicated Smart contract reads and parses CA certificates stored in the blockchain exclusively with the code of Solidity, a compiler for Ethereum smart contract. Notably, as the smart contract does not alter the blockchain, the verification is done for free. The difficulty was connected with the parsing due to absence of Solidity core libraries for string manipulation.

As one can see on Figure 3, although smart contract performance could be less impressive compared to Golang crypto libraries based on the relatively short chains of trusts (less than 400 sub-CAs), starting from chain of trust length exceeding 500 sub-CAs the performance of smart contract based verification is higher than that of Golang code.

For instance, for a chain of trust with the length of around 1100 sub-CAs the smart contract based validation took approximately 7 seconds, while Golang code using Golang cryptographic libraries needed almost 15 seconds. For experiments we used relatively standard DELL workstation with Intel core i7 processor but with the operating memory of 32GB.

B. Costs for initiation of a new CA in Ethereum.

Costs for the support of the infrastructure appears in our view the important advantage of blockchain-based PKI solutions. Assuming local replica of public blockchain to be used by end-users anyway, all costs are basically consist of the fees to the miners that confirm the recording of the data into blockchain.

Based on our experiments with Rinkeby test public blockchain, the initiation of a CA (including creation of empty smart contract, uploading of the certificate into this smart contract, recording of the hash of this certificate into parent CA's smart contract, etc) costs 0,07 Ethers, which based on the present Ether rate of around 700 USD per Ether translates into 50 USD per CA certificate.

Obviously, the initiation of an ordinary end-entity certificate implying only recording of its hash into its parent CA smart contract results in much smaller costs amounting to 7-10 USD per certificate. Given the present price for originating annual end-entity certificate amounting to several hundred USD the blockchain certificate costs seem not to exceed the costs spent on CAs' present infrastructure.

C. Limitation and Challenges

Obviously, one can also find some challenges of blockchain-based PKI infrastructure that comes from the nature of blockchain technology. First, public blockchains are characterized by substantial growth of the blockchain's size replicated to all nodes participating in the system. Especially this is the case for Ethereum and other similar platforms with the support of the smart contracts which are important for arrangement of efficient PKI. For instance, in December 2017, the Ethereum ChainData size with FAST Sync reached 38.89 GB compared to 20.46 GB in September 2017[4].

Second, the extensive volatility of the cryptocurrencies results in some uncertainty with certificate load/update costs in both long run and short term. In other words the cost of blockchain operation is directly linked with the price of the corresponding cryptocurrencies like Ether. For example, in May 2017 the Ether price was 85.43 US dollars[5], while in December 2017 the price reaches 729.01 US dollars which implies the cost of blockchain operations growing 8 times in seven months.

Third, if for certificate validation we use smart contract's parsing with Solidity code rather than external golang module (i.e., Restful services) we can be restricted in the use of hash functions and asymmetric crypto functions available for smart contracts. For instance, for Ethereum without preliminary data processing one can use only SHA256 as hash function and ECDSA signatures based on elliptic curve cryptography.

Lastly, as the access rights to modify certificate authority data is based on user accounts system of the blockchain platform, the lost password results in irrevocably lost access to the account. Some solution of the troubles of CA loosing its blockchain access password could be the arranging empty smart contract and copying all the data from the old smart contract to the new one, as theoretically speaking any smart contract is always available for reading to anyone. Obviously, the establishment of the new CA smart contract could result in reissue of CA certificate (at least in our current implementation), as CA certificates may contain the reference to the corresponding smart contract.

[4]https://etherscan.io/chart2/chaindatasizefast

[5]https://bitcoinmagazine.com/price/

VI. Conclusion

Traditional PKI is CA-based, so the security of PKI systems will be in risk if one CA is compromised. Various researches in the literature investigated using blockchain technology to build secure PKI systems, so they can merge the benefits of the Log-based PKIs and the WoT approaches, and solve some of the problems with conventional PKI system. Therefore our main contribution is a complete working blockchain-based PKI framework. The framework is distributed under the LGPL license and available on Github[6]. The advantages proposed by our framework mitigate the problems with traditional PKI such as the difficulties with rapid certificate revocation, elimination of single points-of-failure and CAs misbehavior. Evaluation results show the benefits of using blockchain technology to build robust PKI system due to reasonable performance and attractive maintenance costs. As future work, we plan to develop a browser plugin to validate certificates based on our blockchain-based PKI framework.

References

[1] J. Yu and M. Ryan, "Evaluating web pkis," in *Software Architecture for Big Data and the Cloud*. Elsevier, 2017, pp. 105–126.

[2] D. Cooper, "Internet x. 509 public key infrastructure certificate and certificate revocation list (crl) profile," 2008.

[3] H. Anada, J. Kawamoto, J. Weng, and K. Sakurai, "Identity-embedding method for decentralized public-key infrastructure," in *International Conference on Trusted Systems*. Springer, 2014, pp. 1–14.

[4] J. Prins and B. U. Cybercrime, "Diginotar certificate authority breach'operation black tulip'," 2011.

[5] B. Laurie, A. Langley, and E. Kasper, "Certificate transparency," Tech. Rep., 2013.

[6] S. Matsumoto and R. M. Reischuk, "Ikp: Turning a pki around with blockchains." *IACR Cryptology ePrint Archive*, vol. 2016, p. 1018, 2016.

[7] S. Matsumoto, P. Szalachowski, and A. Perrig, "Deployment challenges in log-based pki enhancements," in *Proceedings of the Eighth European Workshop on System Security*. ACM, 2015, p. 1.

[8] K. Lewison and F. Corella, "Backing rich credentials with a blockchain pki," 2016.

[9] L. Axon and M. 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.*, 2017, pp. 311–318. [Online]. Available: https://doi.org/10.5220/0006419203110318

[10] C. Fromknecht, D. Velicanu, and S. Yakoubov, "Certcoin: A namecoin based decentralized authentication system," *Massachusetts Inst. Technol., Cambridge, MA, USA, Tech. Rep*, vol. 6, 2014.

[11] "Mozilla included ca certificate list," 2017.

[12] E. Androulaki, C. Cachin, A. D. Caro, A. Sorniotti, and M. Vukolic, "Permissioned blockchains and hyperledger fabric," *ERCIM News*, vol. 2017, no. 110, 2017. [Online]. Available: https://ercim-news.ercim.eu/en110/special/permissioned-blockchains-and-hyperledger-fabric

[13] A. J. Nicholas Stifter and E. Weippl, "A holistic approach to smart contract security," *ERCIM News*, vol. 2017, no. 110, 2017. [Online]. Available: https://ercim-news.ercim.eu/en110/special/a-holistic-approach-to-smart-contract-security

[14] M. Ali, J. C. Nelson, R. Shea, and M. J. Freedman, "Blockstack: A global naming and storage system secured by blockchains." in *USENIX Annual Technical Conference*, 2016, pp. 181–194.

[15] M. Alicherry and A. D. Keromytis, "Doublecheck: Multi-path verification against man-in-the-middle attacks," in *Computers and communications, 2009. iscc 2009. ieee symposium on*. IEEE, 2009, pp. 557–563.

[16] "Blockchain & cyber security. let's discuss," 2017. [Online]. Available: https://www2.deloitte.com/content/dam/Deloitte/ie/Documents/Technology/IE_C_BlockchainandCyberPOV_0417.pdf

[6]https://github.com/snt-sedan/pki-blockchain