Semi-centralized Blockchain-based PKI

Tommaso Peletta
University of Geneva
{tommaso.peletta@etu.unige.ch}

May 16, 2021

Keywords Public Key; PKI; Asymmetric Cryptography; Blockchain; Decentralized; Certificate Authorities; Web of Trust;

Contents

1 Introduction

Public Key Infrastructure (PKI) is a set of processes and technological means that allow the authentication of identities on the Internet. It defines the policies and procedures needed to issue, update, revoke, validate and distribute digital certificates used to secure public key in asymmetric cryptography [2]. The most used certificate standard is the X.509, it is defined as a data structure that binds public key values to subjects. The certificates are issued and signed by trusted certificate authorities (CA), allowing the correct use of public-key encryption. The CA has to strongly validate the identity of the private certificate holder before this assertion. The main goal of the PKI infrastructure is to build a secure network where entities' public keys can be trusted thanks to the validity of the certificate. The infrastructure fails if false certificates are issued and if revocation mechanism are weak or slow.

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. During this event more than 500 fake certificates were discovered [4].

Different centralized solution, such as certificate revocation lists and online certificate status protocol, are specially designed to address these issues, but they don't solve them completely. One recent approach that has been proposed and has been widely studied is a decentralized networks based on a Web of Trust (WoT) concept. In this solution, public-key entity pairs can be verified and certificates can be issued by any trusted entity, or group of trusted entities, in the network. This system removes any central point of failure. However, it makes it difficult for a trusted member of the network to verify the identity of a new user [8].

Blockchain-based PKI can merge the benefits of the Log-based PKIs and the WoT approaches. This solution resolve the potential point of failure of the log-based PKI and facilitate the verification of trusted entities in the WoT.

In this work we briefly analyse current PKI solution based on a blockchain, then we propose an hybrid protocol that combine decentralized verification of entities and central orchestration of infrastructure's processes. The goal of this work won't be to describe a complete and perfect protocol, but to implement the bases of a new infrastructure and theoretically explore further solution and improvement that can be build on top of that.

2 Related Work

In this chapter we summarize some relevant blockchain-based Public Key infrastructure solution. At First, we describe a decentralized solution based on a WoT consensus method and a custom blockchain. Then, we analyse a semi-centralized solution cloud and blockchain-based. Finally, we study a decentralized PKI specifically designed to be compatible with IoTs devices.

2.1 Decentralized Dynamic PKI

Mohnsen Toorani and Christian Gehrmann describe a decentralized solution for a public-key infrastructure based on a custom blockchain in the paper titled *A Decentralized Dynamic PKI based on Blockchain* [7]. Their proposal is decentralized and removes the need for having CAs and CRLs, supporting important functionalities for enrollment, revocation, update, and verification of public-keys. The enrolment and revocation of certificates is done by a WoT process and they are based on consensus between n number of nodes. The consensus mechanism follow the Practical Byzantine Fault Tolerance protocol [1]. All the public keys are validated during the enrolment procedure. Any entity that is already part of the verified entities can initiate a revocation procedure if it detects any malicious activity or invalid key. In order to optimize the verification process, the authors proposed to integrate a dynamic Merkle tree-based accumulator [5].

The described PKI includes three different entities, each of them characterized by a trust weight. The Root units are assumed to be honest entities from the beginning, they are in charge of initiating the blockchain and the first rounds of the consensus protocol. Intermediate units are entities already verified in the PKI, they have a constant trust weight of 1 and can participate in enrollment, update and revocation procedures. Ordinary units are entities that are not part of the verified nodes yet, and they are supposed to be dishonest.

Each block of the custom blockchain includes the timestamp, the Block ID, the accumulator and witness, the tuples (entity ID, pk, flag, role), all the signatures and some other metadata.

To summarize, the selected paper introduce a completely decentralized PKI based on a WoT model. The protocol allow the issuing, revocation and update of certificates using a PBFT consensus mechanism. For efficient verification of public keys, a dynamic cryptographic accumulator is incorporated making the protocol suitable for IoT applications.

2.2 Cloud Blockchain-based PKI

Integrating both blockchain and cloud technologies with a public key infrastructure system can secure certificate authorities and make them resilient against common attacks, that's why Brian Khieu and Melody Moth proposed a semi-centralized PKI based on these two technologies in their paper "CBPKI: Cloud Blockchain-based Public Key Infrastructure" [3].

CBPKI consists of a stateless certificate authority that uses smart contracts to store certificate data on the Ethereum network. The certificate authority is implemented as a Restful API. Once the CA receive a certificate service request, the new certificate is generated and added to the blockchain. Upon revocation of a certificate, the certificate is similarly embedded within a different blockchain for further verification purposes.

The conversion of the certificate authority into a stateless protocol hosted on a cloud platform significantly reduces the size of viable attack surface. More precisely, this solution lowers the value of targeting the CA for attacks and it allows automatic scaling and load-balancing to mitigate DOS attacks.

The paper concludes with a performance and a qualitative evaluation of the CBPKI solution using smart contracts compared to the block storage solution and the traditional PKI model. The smart contracts version allows faster verification time compared to the other two models. Compared to smart contracts, block storage solution requires a traversal across the blockchain in order to find the specific certificate making it slower. Smart contracts are also mined faster and they are cheaper compared to blocks. On the other hand, certificate issuance cost for the CBPKI solution needs to take in account CA, cloud and blockchain costs, making it more expensive than the traditional PKI model.

To summarize, a semi-centralized method based on CA, cloud and smart contracts certificates storage gives a significant security and performance increase over traditional PKIs, on the other hand these improvements come with a higher cost opening the field to further studies in order to mitigate this problem.

2.3 Blockchain-based PKI solution for IoT

The paper *Blockchain-based PKI solution for IoT* [6] analyse the effectiveness of the traditional CA-based PKI on IoT devices and propose some blockchain-based solution to address the main challenges.

The number of Internet of Things (IoT) devices connected in 2020 was estimated to 30 billions and it is rapidly growing. Securing the enormous amount of data generated and transmitted by these devices while minimizing the computational effort and the storage needed is an enormous challenge. One of the key component of the security infrastructure is certificates management. Current CA-based PKI faces numerous problems in the context of IoT.

- **Need to trust manufacturer generated certificates**: Due to the difficulty of interacting with the device for an end-user, manufacturer generally apply directly for certificate the device. This expose the private key of the device to the manufacturer, who may store it for malicious uses.
- **High certificate signing cost**: A single certificate cost between 100$ and 1000$ on average depending on the type of certificate required. When talking about billions of devices this is a real problem for guaranteeing valid and trust-full certificates.
- **Slow certificate signing process**: Traditional CA-based PKI can take several days in order to issue a valid certificate.
- **Difficulties in maintaining certificate lists**: It is very difficult to keep the root certificates updated because these device may not have a user interface.
- **High certificate verification time**: The certificate verification process can often take multiple round-trips, significantly increasing the time taken by the online certificate status protocol check.

The authors of the paper present a high level overview of the working process of their blockchain-based PKI:

1. A device manager generate a pair public-key/private-key.

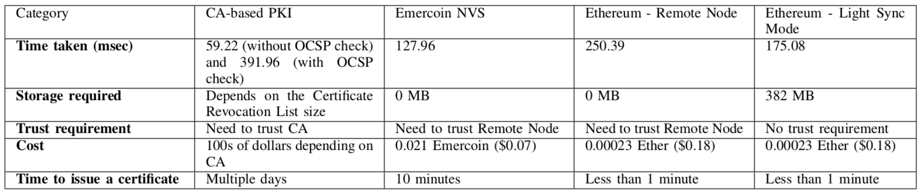
2. The generated public key is used to create a public key certificate.
3. A new ID is generated and it is stored in the blockchain with the certificate hash.
4. the serial ID can be communicated to the recipient and used to query the certificate.

The proposed protocol has been implemented by the authors of the paper with three different approaches. The first one is based on the Emercoin blockchain. Emercoin provides a service to store name-value pairs in its blockchain. Every pair has an owner specified by an Emercoin address stored together with the pair. Only the owner of the address is able to modify or delete the pair.

The second approach is build on the Ethereum blockchain. Ethereum enables smart contracts providing all the basic object oriented programming functionality including the creation of complex data structure and mappings.

The last solution uses the Ethereum Light Sync mode, making it feasible for smaller devices with limited hardware resources to interact with the Ethereum blockchain. This allow light clients to download subset of the block headers and fetch everything else on-demand for the blockchain network. This require more bandwidth for the device.

The evaluation of these three approaches it is summarized in the table in Fig. 1.



Category	CA-based PKI	Emercoin NVS	Ethereum - Remote Node	Ethereum - Light Sync Mode
Time taken (msec)	59.22 (without OCSP check) and 391.96 (with OCSP check)	127.96	250.39	175.08
Storage required	Depends on the Certificate Revocation List size	0 MB	0 MB	382 MB
Trust requirement	Need to trust CA	Need to trust Remote Node	Need to trust Remote Node	No trust requirement
Cost	100s of dollars depending on CA	0.021 Emercoin ($0.07)	0.00023 Ether ($0.18)	0.00023 Ether ($0.18)
Time to issue a certificate	Multiple days	10 minutes	Less than 1 minute	Less than 1 minute

Figure 1: Evaluation of PKI approaches for IoTs [6]

In addition to the table's results, this blockchain-based solution addresses other shortcomings of the traditional CA-based PKI. First of all there is no more need for a centralized TTP. Second it is more resistant to DDoS attacks due to the high number of peers in the blockchain network. Finally there is no need for a separate framework for monitoring and auditing the behaviour of the multiple CAs.

3 Proposed Solution and Implementation

In this section we propose a prototype of a semi-centralized blockchain-based Public Key Infrastructure implemented in python from scratch.

The main idea is to implement a decentralized WoT consensus protocol for the enrolment, update and revocation of public keys while keeping a central authority to orchestrate and manage the requests, the storage and the distribution of information about public keys. This method will resolve the potential point of failure specific of the centralize CAs-based PKI. In addition, keeping a central role helps to optimize the time taken for several procedures such as key verification. A customized blockchain will be used to store e-mail/public-key pairs and all the information needed to validate the corresponding block.

3.1 Client-Server Architecture

Our implemented architecture is based on a Client-Server network, the server act as the central authority in our protocol while clients represent any possible entities entering the network for requesting certificates, authenticating in the network, or verifying public keys.

Every time a client connects to the server, the server sends a challenge to the client. Afterwards, the client can request one of the following action to the server:

- **Verify a pair e-mail/public-key.** Even if the blockchain is public, any client will have the possibility of asking to the server to verify a specific authentication pair. This will remove the need for IoTs devices to download the the blockchain. More over, servers could have specific hardware to speed-up the verification process on the blockchain.
- **Authenticate to the server.** If the client public key is already stored in the blockchain, then by sending a signed message to the server including his pair e-mail/public-key and the challenge sent by the server it will be authenticated in the network until his next disconnection. An authenticated node in the network will be an eligible node for the consensus protocol that will be described later.
- **Certificate his public key.** If a signed message including the pair

e-mail/public-key and the challenge sent by the server is sent by the client to the server and this pair is not yet stored in the blockchain, then a enrolment procedure will be initialized by the server in order to certify the corresponding pair.

3.2 Blockchain Structure

As mentioned before, we implemented a customized blockchain in order to store verified public keys. Each block of the blockchain includes the following elements:

- **Timestamp**
- **Block ID**
- **List of Public Key, E-mail, Challenge, Signature tuples.** A block is implemented to contain three new entities public key to be verified and included in the chain. For each entity there are his public key, his email, the challenge sent by the server at the time of the request and the signature of the new entity that proves the possession of the private key.
- **The Block Hash.** The SHA-256 hash is made by hashing all the data listed above plus the group of consensus emails and the hash of the previous block.
- **The consensus group emails.**
- **The consensus group signatures.** The consensus group sign the hash of the block if they verify the validity of the new public keys pairs. Otherwise, if the list of new public keys is not valid, the consensus group return the hash of the current block.

The first block includes the server public key.

3.3 Consensus Mechanism

Even if the protocol is based on a central server, the issuing of new certificate can't be done without a decentralized WoT consensus mechanism. Once a unverified block of the chain reach 3 new entities that want to certificate their identities, the server randomly choose 3 authenticated nodes connected to the network that will act as the consensus group. Then, the server send to the consensus group the current block. Each entity of the consensus group verify the validity of the current block and send back an answer to the server. More precisely, if the block is valid the answer will be the signature of the block hash, otherwise the answer will be the current block hash. Once received the reply from the consensus group and after correctly verified the correct provenance of the response, the server add the block to the blockchain if more than half of the consensus answer are valid, and it refuse the block otherwise.

4 Further discussion

The public key infrastructure described above does not represent a solution that can be used in a real scenario. More precisely, functional requirements such as revocation and key updating miss in the protocol, and security issues can soars especially if the server credentials are compromised. In this section we will propose solution for some of the major problems of the implemented protocol in order to make it secure and complete.

4.1 Missing Functional Requirements

In order to make the PKI complete 3 important functionalities misses and needs to be implemented:

- **Certificate Revocation:** When a pair of keys is compromised or a dishonest entities is detected, public key certificate needs to be revoked. To do this a flag can be added to each pair email/key in the blockchain, describing the validity or not of the pair. Once a valid key needs to be revoked the same tuple will be added to the next block of the blockchain with the corresponding invalid flag. The verification of public keys will be done starting form the most recent block until the first instance of the email we are looking for is found.
- **Public Key Update:** Similarly to the revocation, the new email/key pair will be simply added to the next block and the verification will be done from the most recent block.

- **Email Verification:** In order to validate a new email/key pair the server and/or the consensus group needs to verify not only the possession of the corresponding private key, but also the possession of the email. This is a major issue in a decentralized PKI. The first solution could be to make the server verify the email, this would make the protocol even more centralized, loosing some of the utility of the decentralized consensus mechanism. The second method is to make the consensus group verifying the emails, meaning that each entity of the consensus group would need to send an email to the node who want to be added to the chain. This would result in a spam of emails that would lead the end user to prefer classical CAs PKI over this solution. Since I didn't find a satisfying solution for this problem I will let it open to further research.

4.2 Security Hotspots

The following security problems soars if the server that manages and orchestrates the infrastructure is compromised:

1. **Denial of Service.** If the only server fails to work as intended, the whole infrastructure fails to work as well. To solve this multiple servers needs to be included in the PKI.

2. **Dishonest Group of Consensus Choice.** If an entity asks to a compromised server to validate his key, the server could choose a defined group of dishonest entities between the connected authenticated nodes in the network, leading to a potential validity of a non-valid block or vice-versa. In order to solve this issue, the consensus group must be chosen by 3 or plus different servers. In addition, the first server must select the other 2 or more randomly. To make sure of the random choose, a modulo function can be applied to the current block hash and the results will indicate which server identifier to choose.

3. **Incorrect Key Verification.** If a client ask to a compromised server to verify an email/key pair the answer could be wrong. A good practice for the client would be to ask the verification to at least 2 different servers, making it very unluckily that both are compromised at the same time.

4. **Authentication to a Compromised Server.** A malicious entity could authenticate on a compromised server without correct credential, making

it potentially part of the consensus mechanism later. The problem is solved by keeping the authenticated node eligible for the consensus group local to each server. This way even if a server has authenticated only dishonest entities it still needs to choose 2 or more extra servers (as explained in point 2) in order to start the consensus mechanism. In this case, external servers will not share the same list of eligible nodes making the malicious entities of the first server be at maximum a third of the total number of chosen entities in the consensus group.

4.3 Evaluation of the Complete Protocol

The proposed implemented public key infrastructure enriched with the solutions discussed above brings clear improvements to the traditional CAs PKI. First of all, it removes the central point of failure characterized by a compromised authority. The validation time of a block depends mostly on the time to verify the email that is depended on the client rapidity. If a client need a new certificate rapidly it will just need to instantly verify the received verification emails, the consensus mechanism validation time is negligible in relation to it.

The cost for certificate issuing depends on both the server orchestration price and the consensus mechanism price. More precisely, each authenticate node is rewarded for each correct block validation done. Normally, the cost of block validation in a WoT mechanism is very low, and since the serves don't need anymore to do all the validation process their cost should be lower than the traditional CAs PKI cost. Since this problem is not entirely part of my expertise domain i will not give precise price and comparison but I will leave it for further research.

The time for verifying a public key validity can be lowered via software and hardware optimization by the server. On top of that, no storage is need for an entity in order to verify data on the blockchain, making the protocol suitable for IoT devices.

The semi-centralized structure of the protocol ensures an easier transition from the traditional model compared to a fully decentralized method. In our case CA could become directly the servers responsible, keeping their company alive even after the transition.

The monetary reward given to authenticated nodes for the validation of blocks would ensure a good load balance of those nodes between the servers (following section 4.3 point 4 solution).

5 Conclusion

Using blockchains for PKIs is promising. In this work, after analysing some promising blockchain-based solution, we proposed an implementation of a semi-centralize blockchain-based PKI. We also discuss the problems related to this model and we proposed solution in order to make it a good alternative to the traditional CAs PKI. The evaluation of the solution shows some promising improvements to the traditional protocol, however some major problems like emails verification aren't yet solved efficiently keeping the topic opened for further researches.

References

[1] M. Castro and C. Liskov. Practical byzantine fault tolerance. *Proceedings of the Third USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 173–186, 1999.

[2] D. Cooper. Internet X.509 public key infrastructure certificate and certificate revocation list(crl) profile. RFC 5280, RFC Editor, 2008.

[3] B. Khieu and M. Moh. Cbpki: Cloud blockchain-based public key infrastructure. *ACM SE '19: Proceedings of the 2019 ACM Southeast Conference*, pages 58–63, 2019.

[4] J.R. Prins. Diginotar certificate authority breach "operation black tulip", September 2011.

[5] L. Reyzin and S. Yakoubov. Efficient asynchronous accumulators for distributed pki. *Security and Cryptography for Networks - 10th International Conference, SCN 2016, Amalfi, Italy*, page 292–309, 2016.

[6] Ankush Singla and Elisa Bertino. Blockchain-based pki solutions for iot. In *2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)*, pages 9–15, 2018.

[7] M. Toorani and C. Gehrmann. A decentralized dynamic pki based on blockchain. *SAC '21: Proceedings of the 36th Annual ACM Symposium on Applied Computing*, pages 1646–1655, 2021.

[8] J. Yu and M. Ryan. Evaluating web pkis. *Software Architecture for Big Data and the Cloud*, 7:105–126, 2017.