Revisiting Privacy-aware Blockchain Public Key Infrastructure

Olamide Omolola and Paul Plessing

Institute of Applied Information Processing and Communication
olamide.omolola@iaik.tugraz.at, plessing@student.tugraz.at

Abstract. Privacy-aware Blockchain Public Key Infrastructure (PB-PKI) is a recent proposal by Louise Axon (2017) to create a privacy-preserving Public Key Infrastructure on the Blockchain. However, PB-PKI suffers from operational problems. We found that the most important change, i.e., the key update process proposed in PB-PKI for privacy is broken. Other issues include authenticating a user during key update and ensuring proper key revocation.
In this paper, we provide solutions to the problems of PB-PKI. We suggest generating fresh keys during key update. Furthermore, we use ring signatures for authenticating the user requesting key updates and use Asynchronous accumulators to handle the deletion of revoked keys. We show that the approach is feasible and implement a proof of concept.

Keywords: Blockchain · Public Key Infrastructure · privacy · RSA.

1 Introduction

Nowadays, Public Key Infrastructure (PKI) plays a major role in ensuring secure communication. PKI works on the principle that a trusted third-party organization called Certificate Authority (CA) can sign certificates and vouch for the authenticity of the link between the public key and the subject name contained within the certificate. The trusted third-party verifies a client's identity and confirms the client's identity before signing. This third-party signs a certificate with its private key[1]. The certificate of the trusted third-party is assumed to be widely known, and another entity can verify the signed client certificate by verifying the signature of the trusted third-party organization affixed to it.

However, PKI is a centralized infrastructure and attacks like those carried out on Comodo[2] or DigiNotar[3] compromises the CAs, and that can compromise the integrity of the certificates issued thereby compromising the whole infrastructure.

Phil Zimmerman proposed the Web of Trust (WoT) in 1992 [15] as a decentralized alternative to PKIs. The initial processes leading to trust in WoT is

[1] The public key is included in the certificate
[2] https://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html(last accessed on 20/03/2019)
[3] https://security.googleblog.com/2011/08/update-on-attempted-man-in-middle.html(last accessed on 20/03/2019)

different from that in PKI. For example, if Alice knows the public key of Bob and trusts him, and Bob knows the public key of Claire, then Alice can ask Bob for Claire's key and trust that it is indeed Claire's key. Alice and Bob exchange their keys initially in person. This exchange event is called a Key Signing Party, where people exchange their keys with each other in person. However, this personal exchange poses two problems. The first problem is an efficiency problem as only a few keys can be exchanged initially in a particular time frame. The second problem is a trust chain problem as it is possible that Alice is unable to find a trust chain that connects with Claire, thereby giving rise to isolated trust communities.

Researchers are continually trying to improve the two mechanisms above in different ways. Blockchain has recently come to the center stage of the research community [10], and many researchers have proposed using the Blockchain to solve some of the PKI and WoT problems. One of such proposals is the Privacy-aware Blockchain-based PKI (PB-PKI) [1]. PB-PKI[4] aims to use the Blockchain to solve the problems of WoT and also ensure privacy.

However, some of the goals were not achieved, and we discovered some problems with the key update process and the key revocation in the proposal.

Our contributions in this paper include the following:

1. **We show that the key update process in PB-PKI [1] does not ensure privacy.**
2. **We propose the use of ring signatures to solve the problem of authenticating registered members of the blockchain during key update to ensure that only registered members can perform key updates.**
3. **We propose a revocation mechanism that involves key deletion from the blockchain for PB-PKI.**
4. **We implement a PoC and show that it is feasible in practice.**

The rest of the paper is structured as follows: Section 2 gives a short overview of previous research; Section 3 gives a short introduction into Asynchronous accumulators and Ring Signatures; Section 4 describes the privacy notions in PB-PKI; Section 5 describes PB-PKI; Section 6 discusses the problems of PB-PKI and provides our solutions to them; we evaluate our ideas in Section 7; Section 8 discusses the implications of our changes to PB-PKI and we conclude in Section 9.

2 Related Work

Blockchain became popular in 2009 with the introduction of Bitcoin [10]. Blockchains are decentralized and store transactions between parties. All the transactions are publicly auditable by all participants, and once a transaction is recorded and confirmed, it is practically immutable. These characteristics have led researchers

[4] PB-PKI is actually an implementation of WoT on the Blockchain

to propose implementing PKIs and WoT on the Blockchain [4, 6–8, 13, 14]. One example of such a proposal is Certcoin [4]. Certcoin is a blockchain variant built upon Namecoin[5] that also functions as a WoT.

The simplest version of Certcoin uses Namecoin as a bulletin board where blockchain posts and blockchain traversals support its functionalities. Data structures such as Asynchronous Accumulators [11] and Kademlia Distributed Hash Table (DHT) [9] were introduced in successive versions so that Certcoin is time- and space-efficient. However, Certcoin was not built with privacy taken into consideration.

Certcoin transactions store information about public key events. The public key events are registration, update, revocation, and verification. Certcoin mandates every entity to register two key pairs. The entity uses the first key pair called online key pair for communication and uses the second key pair called offline key pair for security purposes such as key revocation[6] and update. The offline key pair is stored offline to protect it.

The authors of Certcoin provide an incomplete implementation of Certcoin in the language Go. The implementation uses RSA keys and the Asynchronous Accumulator for key verification. PB-PKI builds on the foundation of Certcoin.

3 Preliminaries

This section briefly describes the algorithms that are used in the rest of the paper

3.1 Asynchronous Accumulator

Reyzin and Yakoubov proposed the Asynchronous accumulator(AA) [11]. AA is the dynamic form of the accumulator called Merkle tree [5], and it provides algorithms for adding or deleting elements from the original set. AA (see figure 1) is a container of several Merkle roots, i.e., an array of Merkle roots. Every index of the AA can contain only the root of one Merkle tree at a particular time.

AA preserves efficiency in an environment, where modifications of a Merkle tree happen often due to frequent addition of leaves. Recalculating the whole tree after a modification requires $\log_n$ time, therefore, it is desirable to optimize this process. AA works similar to a Merkle tree regarding verification.

1. Initially, the AA is empty
2. To add a message m_1, m_1 is hashed to H_1 and put into the first slot of the AA at index 0.
3. Suppose a second message m_2 needs to be added. Since index 0 is occupied by H_1, H_1 and H_2[7] are concatenated and hashed to H_{12}. The hash H_{12} is put at index 1 of the AA. Simultaneously, H_1 gets removed from index 0. To

[5] Namecoin is a fork of Bitcoin
[6] The offline key pair can revoke the online key by signing a revoke-event
[7] H_2 is the hash of message m_2

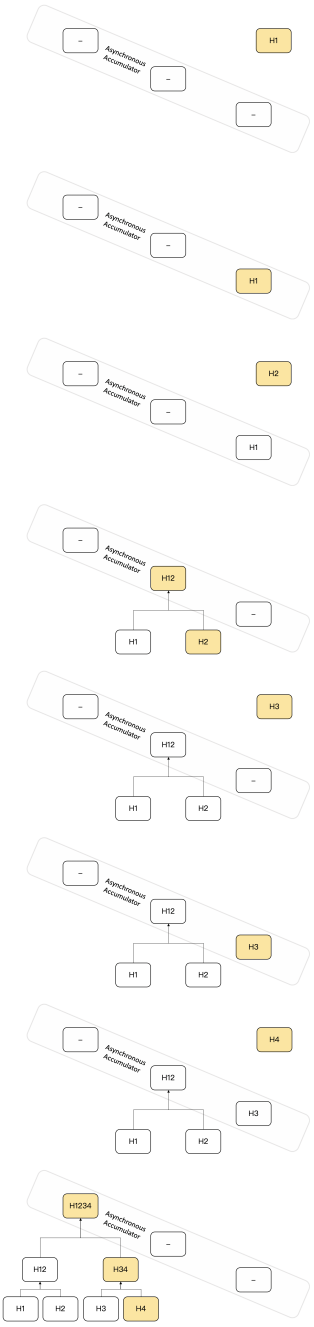

Fig. 1. Illustration of an Asynchronous Accumulator with its empty ("-") and filled ("H****") Merkle roots and their corresponding Merkle trees.

verify that message m_1 is part of the AA, the prover has to remember m_1 and H_2. With this knowledge, he can reconstruct the Merkle root at index 1.
4. When a third message m_3 arrives, it is hashed to H_3 and placed at index 0 since index 0 is empty.
5. The next message m_4 is also hashed to H_4. However, index 0 contains H_3. Therefore, H_3 and H_4 are concatenated and hashed into H_{34}. H_{34} should be placed at index 1, but it is occupied by H_{12}. Thus, H_{12} and H_{34} are concatenated and hashed into H_{1234}. H_{1234} is then placed at index 2. To verify the existence of message m_1, only message m_1, H_2 and H_{34} need to be remembered. Simultaneously, H_3 and H_{12} are removed from index 0 and index 1.

The worst case scenario for an addition to AA is $\log_n$ operations (concatenating and hashing), with n leaves of the biggest tree. In the best case, the addition can be done after one hash if index 0 is free. The trade-off of faster additions is that more space is needed for storing an AA compared to one Merkle root. The storage requirement of AA for n leaves is $\log_n$.

3.2 Ring Signatures

Ring signatures allow a user to sign a message and specify a set of possible signers without revealing which member actually signed the message [12]. The user can choose any set of possible signers that includes himself and sign by using his secret key and the other's public keys without getting their approval or assistance. A ring signature does not need any set-up, and the ring signature scheme is defined by two procedures:

1. **Sign**$(m, P_1, \ldots, P_r, S_s)$ which produces a ring signature σ for the message m, given the public keys $P_1, P_2, \ldots, P_r$ of the r ring members, together with the secret key S_s of the s-th member (who is the actual signer).
2. **Verify**(m, σ) which accepts a message m and a signature σ (the signature includes the public keys of all the possible signers), and outputs either true or false

4 Privacy Concerns

Users sometimes assume that public blockchains are anonymous. This assumption is true to the extent that personal information is not connected to the accounts of the blockchain user. An account is the hash of the public key of a user's key pair on the blockchain. Two possibilities for tracking users exist. For example, it is possible to find out the IP-address of a blockchain address by observing the blockchain network. A user can prevent this possibility by masking his IP-address using anonymizing technologies like Tor. The second violation of privacy happens by design because the public can audit Blockchains. Therefore, the public knows what data is in a blockchain address' wallet, the meta information of the data such as its origin and more. A naive solution would be to use a

new public key for each transaction. However, this has no real privacy gain since the mapping between the public key and the account is available for the public.

In terms of blockchain based PKIs, this violation of privacy would translate to linking any public key and its transactions even when the public key has been updated. At the moment a public key is used in any service, an adversary could track its activities across services. PB-PKI provides unlinkability between an updated public key and its identity without compromising the ability to verify that a specific public key is authorized to take an action.

5 PB-PKI

In this section, we give a short introduction into the original PB-PKI [1].

PB-PKI modifies part of Certcoin to achieve privacy. Registering, revoking and verifying a key in PB-PKI is the same as that of Certcoin. An entity registers its identity by posting its public key on the blockchain. The main difference between Certcoin and PB-PKI is the key update process. The unique key update procedure in PB-PKI aims to provide untrackability and provide a way to disclose the link between an identity and its key by the entity at a later point. This user-controlled disclosure enables the entity to prove that a message is signed with its unlinkable key which is connected to the certified key. PB-PKI hides the link between an identity represented by its current public key and its previous actions as well as keys, while still retaining the authenticity of the key.

Since the PB-PKI Key update process is the main improvement over Certcoin, we will focus on it in the rest of this paper. The PB-PKI Key update process (RSA keys) involves two steps:

1. Generate a new offline key pair, $\mathsf{pkf_n}$ and $\mathsf{skf_n}$ (offline public key at time n and offline secret key at time n), where:
2. Compute the new online key pair $\mathsf{pkn_n}$ and $\mathsf{skn_n}$ (online public key at time n, and online secret key at time n) in the following manner:
   With this formula, the new keys are a valid RSA key pair, where:

$$\mathsf{pkn_n} \cdot \mathsf{skn_n} = 1 \pmod{\mathsf{N_n}}$$

The two steps form a chain of keys as shown in figure 2 after every update. When a user wishes to disclose the public keys, the user has to publish all offline public keys. With that knowledge, anyone can recompute the chain of public keys and verify that a specific key leads back to a particular identity.

Regarding the key update transaction, it must be guaranteed that this new unlinkable public key is from a registered member. Louise Axon [1] proposed that the identity that wants to update must provide a signature signed with the current key. However, this would mean a public linkage of all keys. Therefore, the author proposed that the link is disguised by encrypting the signature with the public keys of a randomly chosen subset of the network members. These network members are then included in the verification process because they can decrypt the signature and verify that it is indeed from a key that was already

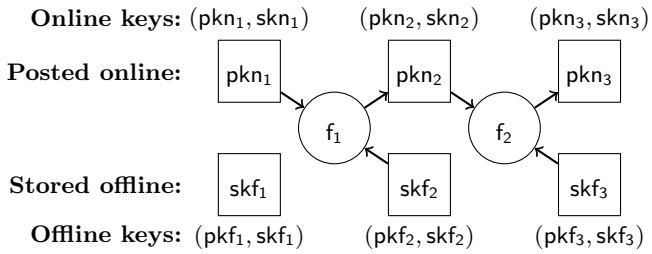

Fig. 2. Key update procedure to facilitate user-controlled disclosure.

on the network. This of course still enables the verifiers to track that two keys belong to the same person. However, since the subset is chosen randomly for every key update, a verifier can only track the identity link between two keys.

Furthermore, the changed usage of the offline key pair means that it cannot be used the same way as in Certcoin. In Certcoin, the offline key provided a way to revoke compromised keys. In case of online key theft, a user can still prevent the malicious usage of the stolen online key by revoking it with the offline key. However, the user stores the offline keys securely after its generation in PB-PKI and only publishes them to disclose its identity. To obtain the same revocation abilities as Certcoin, PB-PKI introduces the so-called Master Key. The Master key functions the same way as offline keys in Certcoin, i.e., it can revoke a current compromised online key.

6 Enhancements to PB-PKI

This section describes the problems encountered in the implementation of PB-PKI and the solutions to these problems[8]. Some of the problems required a total change of procedure while the others needed some modifications to the existing procedures. The issues and their solutions are given below:

6.1 Creation of new Keys

In PB-PKI [1] new keys are generated with the novel update procedure described in section 5. This update procedure should ensure unlinkable keys while allowing user-controlled disclosure at the same time. However, this novel procedure has a side effect - it uses the same modulus for every updated key pair. An attack where encrypted messages can be decrypted when two public keys have the same modulus was documented by Dan Boneh [2]. We present equations below that involves two different public keys $\mathsf{e}_1, \mathsf{e}_2$, and two cipher texts of the same message encrypted by the two public keys A, B. The equations show that the message can

[8] We refer to our improved version of PB-PKI as Enhanced PB-PKI.

Table 1. Comparison of PB-PKI with NOOB-PKI regarding key operations.

PB-PKI	Enhanced PB-PKI
Setup Phase User U generates an online, offline and master RSA key pair.[9]	**Setup Phase** User U generates an online and offline RSA key pair.
Key Registration: U posts a registration transaction with – his identity – a timestamp – the public part of the generated online key – a signature of the generated online public key – a signature of the master key It has to be verified that the identity and the online key have not been registered previously and that the signature of the online key is valid.	**Key Registration:** U posts a registration transaction with – his identity – a timestamp – the public part of the generated online key (registration key) – a signature of the online public key – the witness of the online public key – the public part of the generated offline key – a signature of the offline public key It has to be verified that the identity and the public online key have not been registered previously. The signatures of the online and offline keys and the witness are also validated.
Key Update: User U generates a new offline RSA key pair with public part pkf and private part skf. To generate the new online key with public part pkn and private part skn, U calculates: $\mathsf{pkn}_n = \mathsf{pkn}_{n-1} \cdot \mathsf{skf}_n(\mathsf{ModNn})$ $\mathsf{skn}_n = \mathsf{skn}_{n-1} \div \mathsf{skf}_n(\mathsf{ModNn})$ U then posts an update transaction with – a timestamp – the public part of the newly calculated online key – a signature of the online public key – a signature of the previous online public key to ensure that U is already part of the network. This signature is encrypted with the public keys of a subset of network members. The subset is chosen randomly at each update and has to verify the signature of U's previous online key. U secret shares the new offline public key between a majority of the network members. It has to be verified that the online public key has not been registered previously, that the first signature is valid, and that the second signature is valid and done with a currently valid public key on the network.	**Key Update:** User U generates a new online and offline RSA key pair and posts an update transaction with: – a timestamp – the public part of the new online key – a signature of the new online public key – the witness of the new online public key – the public part of the new offline key – a signature of the new offline public key – a ring signature by use of a randomly selected subset of registration keys and U's registration key – all registration keys used for the ring signature The network members verify that the online public key has not been registered previously, the signatures of the online and offline keys are valid, and the witness is valid. Additionally, the network verifies that the ring signature is valid and that all used registration keys exist.
Key Revocation: User U posts a revocation transaction with – a timestamp – the public part of the to be revoked public key – a signature of the public key Key revocation can be executed either by the key holder.	**Key Revocation:** User U posts a revocation transaction with – a timestamp – the public part of the to be revoked key – a signature of the public key – the witness of the public key – the new ancestors of the revoked public key The network members verify that the revoked key is indeed part of the network and that the signature, the witness, the ancestors are valid. The ancestors of a given message are all its parent nodes in a Merkle tree. The revoked key is replaced with "$\perp$" in the Merkle tree, and the ancestors of "$\perp$" have to be posted so that other users can update their witnesses accordingly.

be decrypted even without the private keys.
Given:

$$\mathsf{A} = \mathsf{M}^{\mathsf{e}_1} \pmod{\mathsf{n}}$$

$$\mathsf{B} = \mathsf{M}^{\mathsf{e}_2} \pmod{\mathsf{n}}$$

If $mathsf\gcd(e_1, e_2) = 1$,there exists some x, y such that $\mathsf{xe}_1 + \mathsf{ye}_2 = 1$
Therefore:

$$\begin{aligned}\mathsf{A}^{\mathsf{x}} \cdot \mathsf{B}^{\mathsf{y}} &= \mathsf{M}^{\mathsf{e}_1\mathsf{x}} \cdot \mathsf{M}^{\mathsf{e}_2\mathsf{y}} \\ &= \mathsf{M}^{\mathsf{xe}_1} \cdot \mathsf{M}^{\mathsf{ye}_2} \\ &= \mathsf{M}^{\mathsf{xe}_1+\mathsf{ye}_2} \\ &= \mathsf{M}^{1} \\ &= \mathsf{M}\end{aligned}$$

Furthermore, an adversary can scan the blockchain for public keys with equal modulus, and be sure that these keys belong to the same user. Instead of user-controlled disclosure, the novel update process actually ensures linkability by design.

Solution As the proposed update procedure could not disguise links between keys, we sought another mechanism. We found that the key update mechanism in PB-PKI is not critical to ensuring privacy. The mechanism helps in disclosure since a user can provide the link between his keys to a verifier. However, there are other ways that one can use to prove the ownership of a key without linking one's keys. One such method is by signing the public key.

We propose generating a new random key pair at each update event. A user can still disclose that a specific key belongs to him. He does it by providing a signature with the key to be disclosed and his registration key. Thus, a verifier can be sure of the ownership of the disclosed key.

6.2 Key Deletion

In previous papers about Certcoin [4] and PB-PKI [1], fast key verification is achieved with the Merkle root of a Merkle tree containing all valid public keys. Specifically, many Merkle roots are involved because Certcoin and PB-PKI use the AA as explained in Subsection 3.1. To quickly verify whether a key is valid, a verifier takes the key as well as its witness and computes its Merkle root. Then the verifier takes the Merkle roots of the latest block and checks whether one of them match the computed one. This process is quick because it needs at most $\mathsf{log_n}$ operations (for n number of keys in the Merkle tree) to calculate the Merkle root. In addition, it is space efficient, because the verifier only needs to store the latest blockheader plus the public key and the witness[10].

[10] Witnesses are the missing hashes needed by a verifier to construct a Merkle root

Adding keys is very efficient as well because of the Asynchronous Accumulator. The maximum of operations needed per addition of a key is $\log_n$ (for n number of all keys in the Merkle tree). Consequently, the Merkle tree does not need to be fully recalculated but it is merged with other trees of the same size, i.e., concatenating the roots and hashing them.

This works very well when one adds keys only. However, the event of a key deletion in case of key revocation was not discussed intensively by the authors of Certcoin and PB-PKI. Even though in both papers it is mentioned that deletion is possible, only Certcoin gives a short explanation of how deletions can be implemented using a Merkle tree. To make the term clear: *deleting* a key from a Merkle tree means to modify an entry of a Merkle tree. The entry to be deleted gets replaced with an entry that indicates that the former public key is deleted.

Solution In the Enhanced version of PB-PKI, we use the sign "$\perp$" for deleted entries. With such a modification, we have to recalculate $\log_n$ parents of the Merkle tree. Therefore, while adding keys takes at worst case $\log_n$ operations, deletion of keys always takes $\log_n$ operations.

Furthermore, the time required for key holders to update their witness needs to be considered as well. Every witness in this Merkle tree has to be updated. The number of operations for witness update is in the worst case in direct correlation to the number of operations needed to update the AA. In the best case, a key holder only has to do one operation to bring his witness up to date. Therefore for both addition and deletion, a key holder needs to do at most $\log_n$ operations.

6.3 Space Efficiency

The second issue regarding entry deletion from Merkle trees is space efficiency. If there are additions to the AA only, the only information that is needed to update any witness are the resulting Merkle roots after every addition. However, in the case of deletion, one needs to know the former witness of the deleted element in order to update the other witnesses of a Merkle tree. This means that the witness of every revoked and deleted key has to be stored and published. The result of this is more storage is used.

Solution The AA changes with every transaction, but only the final accumulator after all transactions in a block are taken into account gets shown. This means that for key additions it is mandatory to publish the witness of each added key as well. The result is the publication of every change of the AA.

As a result of the changes made in the preceding paragraphs, a prover with an old witness has to go back to the last block where his public key plus witness was valid. Then he traverses all transactions of all blocks leading to the most current block, and he adjusts his witness according to the occurred key events. Finally, he has a witness that provides proof when using the latest accumulator.

6.4 User Authentication

The third challenge was the authentication of a pending key update. At key update, it should be ensured that the user requesting the key update already owns a key on the network while securing his identity. In the proposed PB-PKI this is done through a signature of the current key. However, this would enable a linking of the keys. Therefore, the signature is encrypted. It is encrypted with a subset of total public keys available on the network. These members then have to decrypt the signature and verify that it is indeed done with a registered key.

This practice sounds nice in theory but is hard to implement. The first set of challenges arise: how can the subset of network members announce their verification of the signature? Post it on the blockchain? How can one trust that they are telling the truth? Moreover, how can one guarantee that they actually verify the signature and have not been offline for two years?

This set of questions remains unanswered but even if there were answers to them, another question arises: How to find out a subset of current public keys? This would either require global storage of `n` public keys (for `n` network members), like the AA, but instead of a few Merkle roots we needed to record all valid keys, millions of it in every block. This would be fast but space intensive (totally unusable). Alternatively, we traverse the blockchain looking for valid keys, which would be space friendly but time intensive. Additionally, we cannot find out which public keys that were posted onto the blockchain are still valid and which are not, because no keys are linked to each other.

Solution We decided to find another way of solving the authentication problem. We chose to use ring signatures. Ring signatures are explained in Subsection 3.2. In the Enhanced PB-PKI, a ring signature is crafted and posted in the update transaction. The public keys are randomly selected out of the public keys published at registration event. The user's public key that was created during registration is added to the set of keys. With these selected keys, he performs the ring signature and puts it onto the transaction. That way, anyone can verify that this transaction must be from one of the already registered holders of the used public keys. However, nobody knows who it was from this group of keys. Moreover, by carefully selecting different large sets at each update, an attacker cannot tell which key belongs to which identity. We used a ring size of 6 possible signers for the proof of concept because having a higher ring size is space inefficient.

7 Evaluation

For the proof of concept, we used an Asus Laptop with 4GB RAM and an Intel Core processor i5 of the 5th generation. The installed operating system was Ubuntu 16.04. The PoC was written in the programming language Go, and the database we used to store the blocks was LevelDB.

With this setup, we measured how long it takes to execute the different operations:

- Register a Key: ~80 milliseconds
- Update a Key: ~120 milliseconds
- Revoke a Key: ~10 milliseconds
- Verify a Key: ~15 microseconds
- Update a Witness: ~90 milliseconds

Verifying a key is the most critical activity because users often have to verify whether a key is valid. In Enhanced PB-PKI, verifying a key is the fastest operation, because it only involves hashing and concatenation without any write operation. The bottleneck of posting transactions is the need to mine new blocks and verify them. Setting the mining time or process is beyond the scope of this paper. The Enhanced PB-PKI is a full blockchain and it was deployed locally on the laptop for the experiments.

8 Discussion

In this section, we consider the possible cases of key compromise and the options that the Enhanced PB-PKI provides to the victim. Key compromise happens when an adversary gets access to or steals the private part of the online or the offline key. Getting access to a key means an adversary knows a key while the victim still has access to the key too. Stealing a key means an adversary takes the key away from the victim so that the adversary knows the key while the victim does not.

8.1 Security Model

We consider a model where PB-PKI is accessible to the public. In this model, we assume that an adversary has access to the blockchain. The adversary is also part of every operation involving registration, key update and key revocation and able to tamper with any of these operations.

The security goals of the system are as follows:

1. The adversary cannot link a key to the owner.
2. Only the owner of a public online key can prove the ownership.

Depending on which private keys were stolen or accessed, we consider six different cases:

1. **Online secret key accessed only:** The adversary can take part in the network by using the accessed online key. The victim can revoke his online key and update it. The update creates a new online key he can safely use again.
2. **Online and offline secret keys accessed:** The adversary can take part in the network by using the accessed online key. The adversary can also perform key update, but this is not useful because when the user revokes the online key and performs key update, then the adversary can no longer impersonate the victim. After key update, the adversary does not have any knowledge of either the new online or offline key.

3. **Online secret key stolen only:** The adversary can take part in the network by using the stolen online key. To recover from this case, it requires that the victim stored the previous online key. With this previous online key and the current offline key, he can recalculate the stolen online key. Then he revokes and updates the stolen key.
4. **Online and offline secret keys stolen:** The adversary can take part in the network by using the stolen online key. In this case, the victim can do nothing against the adversary. However, the victim can update the previous online key, and stay part of the network. The possibilities for the adversary are limited to the use of the stolen online key only. To mitigate this scenario, the introduction of an expiration date of public keys would limit the key abuse of the adversary to a certain time frame.

As long as keys are not stolen and key compromise is detected early, an adversary can be kept under control. To avoid key theft, keys should always be copied to a safe location.

9 Conclusion

In this paper, we have shown that the initial proposal of PB-PKI is fraught with challenges. Some of the problems include authentication during key update, how revoked keys can be deleted successfully and the key update mechanism is not as secure as earlier expected. Enhanced PB-PKI simplifies the update process by requiring the user to generate fresh keys without using the key updates mechanism by Louise Axon et al. [1]. The new key update mechanism removed the privacy problem that was introduced by PB-PKI . The Enhanced PB-PKI solved the problem of user authentication by introducing ring signatures to PB-PKI. We also adapt the use of AA in order to aid key deletion and ensure that storage space is maximized after deletion of a revoked key. The Certcoin usage of offline keys for key revocation was retained instead of using it for key updates as in the initial PB-PKI construction. We developed the proof of concept using the Go programming language and show that the solutions we proposed are feasible.

In the future, we plan to implement optimizations to the ring signature scheme. New ring signature schemes such as forward-secure linkable ring signatures [3] will also be investigated as this supports forward security and the blockchain user can prove ownership of a ring signature using this scheme.

Acknowledgment

We acknowledge David Derler, Sebastian Ramacher, Peter Lipp and Clemens Brunner for fruitful discussions during the period of this work. We acknowledge Sebastian Ramacher, Peter Lipp and Clemens Brunner for their thorough reviews.

This research is part of the LIGHTest project funded by the European Unions Horizon 2020 research and innovation programme under G.A. No 700321.

References

1. 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 - Volume 6: SECRYPT, (ICETE 2017)*, INSTICC. SciTePress, 2017, pp. 311–318.
2. D. Boneh, "Twenty years of attacks on the rsa cryptosystem," 1998.
3. X. Boyen and T. Haines, "Forward-secure linkable ring signatures from bilinear maps," *Cryptography*, vol. 2, no. 4, p. 35, 2018. [Online]. Available: https://doi.org/10.3390/cryptography2040035
4. S. Y. Conner Fromknecht, Dragos Velicanu, "Certcoin: A namecoin based decentralized authentication system," 2014.
5. D. Derler, C. Hanser, and D. Slamanig, "Revisiting cryptographic accumulators, additional properties and relations to other primitives," in *Topics in Cryptology - CT-RSA 2015, The Cryptographer's Track at the RSA Conference 2015, San Francisco, CA, USA, April 20-24, 2015. Proceedings*, 2015, pp. 127–144. [Online]. Available: https://doi.org/10.1007/978-3-319-16715-2_7
6. J. A. Garay, A. Kiayias, N. Leonardos, and G. Panagiotakos, "Bootstrapping the blockchain, with applications to consensus and fast PKI setup," in *Public-Key Cryptography - PKC 2018 - 21st IACR International Conference on Practice and Theory of Public-Key Cryptography, Rio de Janeiro, Brazil, March 25-29, 2018, Proceedings, Part II*, 2018, pp. 465–495. [Online]. Available: https://doi.org/10.1007/978-3-319-76581-5_16
7. E. Karaarslan and E. Adiguzel, "Blockchain based DNS and PKI solutions," *IEEE Communications Standards Magazine*, vol. 2, no. 3, pp. 52–57, 2018. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/MCOMSTD.2018.1800023
8. S. Matsumoto and R. M. Reischuk, "IKP: turning a PKI around with blockchains," *IACR Cryptology ePrint Archive*, vol. 2016, p. 1018, 2016. [Online]. Available: http://eprint.iacr.org/2016/1018
9. P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the xor metric," in *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, ser. IPTPS '01. London, UK, UK: Springer-Verlag, 2002, pp. 53–65. [Online]. Available: http://dl.acm.org/citation.cfm?id=646334.687801
10. S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2009.
11. L. Reyzin and S. Yakoubov, "Efficient asynchronous accumulators for distributed pki," *IACR Cryptology ePrint Archive*, vol. 2015, p. 718, 2015.
12. R. L. Rivest, A. Shamir, and Y. Tauman, "How to leak a secret," in *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings*, 2001, pp. 552–565. [Online]. Available: https://doi.org/10.1007/3-540-45682-1_32
13. A. Singla and E. Bertino, "Blockchain-based PKI solutions for iot," in *4th IEEE International Conference on Collaboration and Internet Computing, CIC 2018, Philadelphia, PA, USA, October 18-20, 2018*, 2018, pp. 9–15. [Online]. Available: https://doi.org/10.1109/CIC.2018.00-45
14. A. Yakubov, W. M. Shbair, A. Wallbom, D. Sanda, and R. State, "A blockchain-based PKI management framework," in *2018 IEEE/IFIP Network Operations and Management Symposium, NOMS 2018, Taipei, Taiwan, April 23-27, 2018*, 2018, pp. 1–6. [Online]. Available: https://doi.org/10.1109/NOMS.2018.8406325

15. P. Zimmermann. (1994) Pgp user's guide, volume i: Essential topics. [Online]. Available: https://web.pa.msu.edu/reference/pgpdoc1.html