A Decentralized Dynamic PKI based on Blockchain

Toorani, Mohsen; Gehrmann, Christian

Published in:
Proceedings of the 36th ACM/SIGAPP Symposium On Applied Computing (SAC'21)

DOI:
10.1145/3412841.3442038

2021

Document Version:
Early version, also known as pre-print

A Decentralized Dynamic PKI based on Blockchain

Mohsen Toorani
mohsen.toorani@eit.lth.se

Christian Gehrmann
christian.gehrmann@eit.lth.se

Department of Electrical and Information Technology
Lund University, Sweden

Abstract

The central role of the certificate authority (CA) in traditional public key infrastructure (PKI) makes it fragile and prone to compromises and operational failures. Maintaining CAs and revocation lists is demanding especially in loosely-connected and large systems. Log-based PKIs have been proposed as a remedy but they do not solve the problem effectively. We provide a general model and a solution for decentralized and dynamic PKI based on a blockchain and web of trust model where the traditional CA and digital certificates are removed and instead, everything is registered on the blockchain. Registration, revocation, and update of public keys are based on a consensus mechanism between a certain number of entities that are already part of the system. Any node which is part of the system can be an auditor and initiate the revocation procedure once it finds out malicious activities. Revocation lists are no longer required as any node can efficiently verify the public keys through witnesses.

1 Introduction

PKI has been used for a long time for secure channel establishment and protection of data objects. In the traditional PKI, the CA has a central role and is responsible for enrolling and revoking public keys. This makes the PKI ecosystem fragile and prone to attacks and operational failures: several CAs have been compromised, and many fraudulent certificates were issued and used for man-in-the-middle attacks [1]. Moreover, maintaining CAs, and distribution and storage of certificate revocation lists (CRLs) is demanding, especially in loosely-connected and large systems.

Alternatives to the CA-based PKIs include the web of trust (WoT), identity-based encryption (IBE), and certificateless public key cryptography (CL-PKC). WoT provides a decentralized trust model where the authenticity of public keys and their owners is based on the amount of trust that other entities in WoT have in the entity in question. This removes the need for a CA but has shortcomings in providing non-repudiation, addressing revocation, and the need for a priori trust relations.

Despite problems and limitations in the centralized trust model of the traditional PKI, it is scalable and cost-effective and provides integrity, authentication, and non-repudiation. It is then worth-investigating further improvements to the PKI model than replacing it with its alternatives. Several new PKI models have then been proposed, mainly based on *public logs* or use of *blockchain*:

 A variant will appear in SAC'21, https://doi.org/10.1145/3412841.3442038.

1. **Log-based PKIs**: In this approach which is mainly developed for addressing the problem of misbehaving CAs, certificates are not deemed valid until they are logged on append-only public logs. *Certificate Transparency* (CT) [2] was the first log-based solution, proposed by Google for improving the accountability of CA operations and detecting mis-issued certificates. Anyone can audit CA activities by monitoring CT logs and notice the issuance of suspect certificates that are not in public logs. The main proposal [2], however, does not specify any mechanism for revocation and authenticity of logged certificates, but there are proposals for handling revocation [3]. The idea was then extended for building a transparent PKI as in ARPKI [4] and PoliCert [5], and for providing transparency in messaging systems as in CIRT [3], CONIKS [6], and EthIKS [7]. Such extensions typically require some extra (trusted) entities in addition to the requirement for a public log [8, 4, 9]. Log-based PKIs have several drawbacks: They require a centralized and consistent source of information to operate securely, do not sufficiently incentivize recording or monitoring CA behaviors, and require time and manual efforts for reporting CA misbehavior.

2. **Blockchain-based PKIs**: Blockchain is an append-only database of signed transactions that can be used to eliminate central points-of-failure by providing a decentralized solution and provide certificate transparency and revocation, and reliable transaction records. From the trust model perspective, two approaches have been taken in proposals for blockchain-based PKI:

    - **Hierarchical**: Most proposals for blockchain-based PKI have a hierarchical trust model where some CAs will decide about certification or revocation of keys [10, 1, 11, 12, 13, 14]. They are indeed log-based PKIs that use blockchain as an append-only public bulletin. Some proposals, however, involve a few more entities than only CAs and provide some relaxed centralization but still with limited functionalities since trusted central authorities are not eliminated.
    - **Web of trust**: Blockchain-based PKIs based on WoT [15, 16, 17, 18] typically replace CA with miners in public blockchains such as Bitcoin or Ethereum, and certificates are generated by mining after PoW. Certificate owners typically pay miners for their work and there is no mechanism for preventing or revoking mis-issued certificates. Another approach which is proposed in this paper is to distribute trust between entities, and certification and revocation take place after a consensus between entities.

In this paper, we propose a dynamic blockchain-based PKI based on WoT and without any central role for the CA such that new keys can be securely enrolled or revoked based on a consensus mechanism between trusted nodes that are already part of the system. The possibility to dynamically add and revoke nodes is a requirement in fully-distributed and loosely-connected systems such as smart city and IoT applications. Furthermore, we seek a PKI solution for such systems such that despite the lack of a CA, the participating nodes can easily determine whether or not a particular public key can be regarded as trusted. This resolves a major problem in loosely-connected distributed systems where it is hard to assign a single or a few trusted nodes to enroll and revoke keys in the system. A PKI built on the assumption of the presence of such roles such as the one proposed in [11] will take away the main benefits of using a blockchain-based PKI to a large extent since enrollment and revocation decisions must always go through a traditional CA. Hence, we instead suggest a novel method where the key enrollment and revocation decisions are taken by the nodes that are part of the PKI and not a CA. This allows a sub-part of the system to act completely autonomous from the rest of the system for the key enrollment and revocation but comes at the price of the risk that a sufficient number of malicious nodes might be able to exclude honest nodes from the PKI. We resolve this problem by using the practical Byzantine fault tolerance (PBFT) [19, 20, 21, 22] as the consensus mechanism within the enrollment and revocation procedures. The scheme will then tolerate up to $\lfloor(\mathtt{t}-1)/3\rfloor$ malicious nodes where $\mathtt{t}$ denotes the number of nodes in the consensus group.

Another problem in a blockchain-based PKI is the efficient verification of set membership. In a traditional PKI, such verification could be simply done by verifying the CA's signature in the certificate and

verifying that the certificate is not revoked, for instance, using OCSP [23] or CRL [24]. However, in an ordinary blockchain-based PKI where enrolled and revoked keys are added to the blockchain in chronological order, whenever a node wants to verify that a given public key is valid, it has to read many blocks on the chain and then verifies the corresponding signatures. This would be demanding, especially for resource-constrained nodes. To tackle this problem, we use a dynamic cryptographic accumulator, a space-efficient data structure that supports a fast verification of set membership. This can reduce the required time and space for verification from linear to logarithmic. Cryptographic accumulators were introduced by Benaloh and de Mare in 1993 as a decentralized alternative to digital signatures [25] and have been used in different applications. When an element is added to a set through an accumulator, a witness is generated that can be used later to prove the membership of the element. We use a dynamic accumulator that supports the deletion of elements. Each new block in the proposed solution will contain an updated accumulator and witnesses. For verifying a key, a node will then only need to access the last block in the chain. The previous blocks are stored and available for auditing by anyone and further follow-up of malicious activities. Our contributions can be summarized as:

- We introduce a general model and a solution for dynamic blockchain-based PKI that we call it DBPKI. It is decentralized, removes the need for having CAs and CRLs, and supports important functionalities for enrollment, revocation, update, and verification of public keys.
- We incorporate WoT into blockchain-based PKI such that enrollment and revocation of nodes are based on consensus between `t` number of nodes. We deploy PBFT as the consensus mechanism.
- All the public keys are validated during the enrollment procedure. Any entity that is part of the system can act as an auditor and initiate the revocation procedure if it detects any malicious activity or invalid key.
- For efficient verification of the public keys, we incorporate a *dynamic* Merkle tree-based accumulator into the proposed solution.

1.1 Related work

Since the introduction of the Bitcoin [26], blockchain technology received lots of attention and got many applications such as smart cities [27] and Internet-of-Things (IoT) [28]. The blockchain enables secure and fully-distributed log management which is suitable for peer-to-peer networks. Transaction logs are included in a chain of blocks where each block contains a secure one-way hash of the previous block. A new block is only allowed to be added to the chain if it has gone through a *consensus* decision between the peers in the network. The consensus mechanism is a critical part in blockchain and ensures its security and efficiency. The most utilized consensus mechanism in permissionless blockchain and cryptocurrencies is *proof of work* (PoW), which is very energy consuming [29]. *Byzantine fault tolerant* (BFT) protocols are well-known consensus mechanisms that tolerate malicious faults, but many of them are inefficient to be used in practice or rely on assumptions that can be easily invalidated by an attacker.

Most proposals for blockchain-based PKI simply consider the blockchain as an append-only public bulletin and keep having CA and thus provide very limited functionalities: Wang et al. [10] used blockchain as an append-only public log to monitor CA's certificate signing and revocation operations on TLS certificates. CA-signed certificates and their revocation status are submitted by the web server as a transaction and appended to the blockchain by the miners after verifying transactions. This solution is still dependent on a traditional CA role, and the blockchain mainly serves as a source for certificate status registration and look-up. Kubilay et al. [1] introduced CertLedger which uses public blockchain as a public log to validate, store, and revoke TLS certificates. Lewison et al. [11] used the Ethereum platform for PKI management

and maintaining keys but used a centralized CA model for adding and revoking keys. The blockchain is then used as a public log, and the CA acts as the root of the chain and issues the certificates. Matsumoto and Reischuk proposed IKP [14], an Ethereum-based PKI enhancement that uses smart contracts to offer automatic responses to CA misbehavior and incentives for those who help in detecting misbehaviors but still, CA has a central role and the trust model is hierarchical. Yang et al. [12] proposed BC-PKM, a public key management system based on blockchain for named data networking. It includes basic functionalities that we consider in this paper but does not enjoy our WoT-based mechanism for adding and revoking public keys. Qin et al. [17] proposed *Cecoin* where digital certificates are treated as currencies and stored on a decentralized database where states are updated by miners on Bitcoin-based blockchain. They replace CAs with miners in the Bitcoin so for obtaining a certificate, the certificate owner must pay some cecoins to miners for their contributions and wait until they mine a new block through a PoW mechanism. They deploy a modified Merkle Patricia tree for retrieval and verification of certificates. AlBassam [16] introduced SCPKI where each entity uses smart contracts to publish a set of attributes, signatures, and revocations for its identity on the Ethereum platform and allows entities to store, retrieve and verify identities through a PGP-like WoT. However, it does not provide functionalities and features considered in this paper and has other limitations including adaptability and privacy. Ali et al. [15] introduced *Blockstack* that leverages the Bitcoin blockchain to provide a name registration service that allows users to bind public keys to their names. Their solution was initially based on Namecoin, but after finding some security problems in Namecoin, they migrated to Bitcoin. *Namecoin*, a cryptocurrency forked from the Bitcoin, was introduced for a decentralized DNS for *.bit* addresses, where self-signed TLS certificates of a domain can be added to DNS addresses as auxiliary information. Fromknecht et al. [18] proposed *Certcoin*, a blockchain-based PKI which ensures *identity retention*, i.e. to prevent registering different public keys for one identity. Certcoin is based on Namecoin cryptocurrency and includes two versions: a version that supports efficient verification uses accumulators, and a version that supports efficient lookup is based on an authenticated distributed hash table. Certcoin is based on PoW and mining concept and does not enjoy our WoT and consensus mechanism or a dynamic accumulator that supports deletion. It cannot prevent identity squatting, does not support recoverability for identities that are falsely added to the blockchain, and does not fulfill a user's demand of using multiple public keys under the same identity, which is the case in many applications such as distributed IoT systems. Anada et al. [30] proposed to include not only the subject ID into a public key value but also the guarantors' public key ID. They suggest using blockchain for maintaining the list of keys in a consistent way across the network. However, their approach does not cover the actual principles for maintaining this log but rather focuses on the inclusion of public key IDs into the public key values themselves.

Similar to some other blockchain-based PKI schemes [30, 18, 11], our proposed scheme uses blockchain to maintain the list of all trusted and revoked public keys. However, different from previous work, in the proposed solution a new public key will be accepted and added to the blockchain if and only if the key is confirmed by at least a pre-defined number `t` of trusted nodes that are already part of the system. This number will increase as the blockchain size increases. Similarly, keys can be revoked if at least `t` trusted entities agree. Once a key is revoked, it is not allowed to be added to the chain again. Instead, units that want to join the PKI again will be forced to generate new key pairs and instantiate the enrollment procedure again. A comparison between the proposed scheme and related work is provided in Table 1 where storage type denotes whether full or hash of public keys are stored on the chain.

We do not consider privacy-awareness [32, 33] in this paper, but the DBPKI can be updated to provide it if needed. Privacy-awareness in blockchain-based PKI was considered first in PB-PKI [32] which is the same as the Certcoin in registering, verifying, and revoking public keys, but has a different key update mechanism. Each entity has offline and online pairs of public-private keys and registers its identity by posting its online public key on the blockchain. The main difference with the Certcoin resides in their key update procedure which removes the direct link between identity and public key and introduces an update function that given the new offline private key and old online public key, outputs a new online public key. The solution, however,

Scheme	Centralization	Trust Model	Blockchain Type	Consensus	Certificate format	Updatable key	On-chain storage
CertLedger [1]	Semi-centralized	Hierarchical	Ethereum-based	PBFT	X.509	No	Hash only
Lewison et al. [11]	Semi-centralized	Hierarchical	Ethereum	N/A	Custom	No	Full
Wang et al. [10]	Semi-centralized	Hierarchical	Custom	N/A	X.509	Yes	Full
Yakubov et al. [31]	Semi-centralized	Hierarchical	Ethereum	N/A	X.509v3	No	Full
CBPKI [13]	Semi-centralized	Hierarchical	Ethereum	N/A	X.509	No	Hash only
IKP [14]	Semi-centralized	Hierarchical	Ethereum-based	N/A	X.509	Yes	Full
Blockstack [15]	Decentralized	WoT	Bitcoin	PoW	Custom	No	Hash only
SCPKI [16]	Decentralized	WoT	Ethereum	N/A	Custom	No	Hash only
Cecoin [17]	Decentralized	WoT	Bitcoin	PoW	Custom	Yes	Full
Certcoin [18]	Decentralized	WoT	Namecoin	PoW	Custom	Yes	Full
Proposed scheme (DBPKI)	Decentralized	WoT	Custom	PBFT [1]	Custom	Yes	Hash only

Table 1: Comparison between proposals for blockchain-based PKI
[1] It can be replaced with more scalable and efficient variants of the PBFT.

is criticized in [33] by showing problems in the key update procedure, user authentication during the key update, and key revocation.

2 Preliminaries

Definition 1. A *signature scheme* $SIG = (\mathsf{KG}, \mathsf{Sign}, \mathsf{Veri})$ consists of the following polynomial-time algorithms:

- $\mathsf{KG}(1^\lambda) \rightarrow (\mathsf{sk}, \mathsf{pk})$: The key generation algorithm is a randomized algorithm that takes security parameter λ as input and outputs a pair of private and public keys.
- $\mathsf{Sign}(\mathsf{sk}, \mathrm{m}) \rightarrow \sigma$: The signing algorithm signs the message $\mathrm{m} \in \{0,1\}^*$ with the private key sk and returns digital signature σ.
- $\mathsf{Veri}(\mathsf{pk}, \sigma, \mathrm{m}) \rightarrow 1/\perp$: The signature verification is a deterministic algorithm that determines whether or not σ is a valid signature for m under the corresponding private key of public key pk.

Definition 2. A hash function $\mathsf{H} : \{0,1\}^* \rightarrow \{0,1\}^n$ is cryptographic secure if it is preimage resistant, second preimage resistant, and collision resistant.

Definition 3. A signature scheme $SIG = (\mathsf{KG}, \mathsf{Sign}, \mathsf{Veri})$ is *existentially unforgeable under an adaptive chosen message attack* or EUF-CMA-secure [34] if for any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$, the adversarial advantage in winning the security experiment defined in Figure 1 is negligible, i.e. we have:

$$\mathbf{Adv}_{SIG}^{\mathsf{EUF\text{-}CMA}} = \Pr[\mathbf{Exp}_{SIG}^{\mathsf{EUF\text{-}CMA}}(\mathcal{A}) = 1] \leq negl(\lambda) \tag{1}$$

where $negl$ denotes a negligible function, and λ is the security parameter.

Definition 4. A cryptographic *accumulator* ACC basically consists of four polynomial-time algorithms AccGen, AccAdd, AccWitAdd and AccVer [35, 36]:

- $\mathsf{AccGen}(1^\lambda) \rightarrow \mathrm{a}_0$: given the security parameter λ, initiates the accumulator by an empty set a_0, as well as some additional parameters if needed.

$\underline{\mathbf{Exp}^{\mathsf{EUF\text{-}CMA}}_{SIG}(\mathcal{A})}$
$(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{KG}(1^\lambda)$
$ML \leftarrow \emptyset$
$(\sigma', m') \xleftarrow{\$} \mathcal{A}^{O.\mathsf{Sign}}(\mathsf{pk})$
$t \leftarrow \mathsf{Veri}(\mathsf{pk}, \sigma', m') \wedge (m' \notin ML)$
return t

$\underline{O.\mathsf{Sign}(m)}$
$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$
$ML \xleftarrow{\cup} m$
return (σ, m)

Figure 1: Security experiment for EUF-CMA-secure signature schemes. Adversary $\mathcal{A}$ can adaptively query $O.\mathsf{Sign}$ oracle to sign its chosen messages.

- $\mathsf{AccAdd}(\mathrm{a}_i, x) \rightarrow (\mathrm{a}_{i+1}, w^x_{i+1}, \mathsf{updmsg}_{i+1})$: takes in the current state of the accumulator a_i and the value to be added x, and outputs an updated accumulator value a_{i+1} and the membership witness w^x_{i+1} for x. Additionally, an update message updmsg_{i+1} is generated which can be used by any other witness holders to update their witnesses.
- $\mathsf{AccWitAdd}(w^x_i, y, \mathsf{updmsg}_{i+1}) \rightarrow w^x_{i+1}$: updates the witness for element x after another element y is added to the accumulator. The update message updmsg_{i+1} may contain any subset of $\{x, \mathrm{a}_i, \mathrm{a}_{i+1}, w^y_{i+1}\}$ and other parameters.
- $\mathsf{AccVer}(\mathrm{a}_i, x, w^x_i) \rightarrow 1/\perp$: verifies the membership of x in the accumulator using witness w^x_i and accumulator state a_i.

Any accumulator should provide *correctness* and *soundness* properties. *Dynamic* accumulators [37] that support deletion of elements from the accumulator have the following two additional algorithms:

- $\mathsf{AccDel}(\mathrm{a}_i, x) \rightarrow (\mathrm{a}_{i+1}, \mathsf{updmsg}_{i+1})$: deletes element x from the accumulator.
- $\mathsf{MemWitUpdateDel}(x, w^x_i, \mathsf{updmsg}_{i+1}) \rightarrow w^x_{i+1}$: After deletion of y from the accumulator, it updates the membership witness for element x.

Definition 5. A dynamic accumulator is *sound* (or simply secure) if it is difficult to fabricate a witness w^x for a value x that has not been added to the accumulator [35, 36]. More formally, for any security parameter λ and any stateful PPT adversary $\mathcal{A}$ with black-box access to AccAdd and AccDel oracles which take elements x_0 on accumulator a, we should have:

$$\Pr\left[\begin{array}{l} \mathrm{a}_0 \leftarrow \mathsf{AccGen}(1^\lambda); L \leftarrow \emptyset; \\ (x, w^x) \leftarrow \mathcal{A}^{\mathsf{AccAdd},\mathsf{AccDel}(\mathrm{a}_0, x_0)}; \\ x \notin L : \mathsf{AccVer}(\mathrm{a}_1, x, w^x) = 1 \end{array}\right] \leq negl(\lambda) \tag{2}$$

where $negl$ is a negligible function in the security parameter λ, and a_1 denotes the accumulator state at the end of the security experiment. L (which is initiated with an empty set) keeps the list of elements that are used in adversarial calls, and will be updated after each adversarial call to the AccAdd and AccDel oracles.

Definition 6. We define a *ledger* $\mathbf{x} \in \mathcal{L}$ as a vector of sequences of transactions $\mathrm{tx} \in \mathcal{T}$. Transactions are defined through a digital signature scheme as defined in Definition 1. A transaction tx is of the form $\{\mathsf{ID}_1, \mathsf{ID}_2, \ldots, \mathsf{ID}_\mathsf{t}\} \rightarrow (\sigma, \{\})$ where σ is a vector $\langle(\mathsf{pk}_1, \sigma_1), \ldots, (\mathsf{pk}_\mathsf{t}, \sigma_\mathsf{t})\rangle$ of public keys and corresponding digital signatures. A transaction tx is valid with respect to a ledger if all digital signatures are verified.

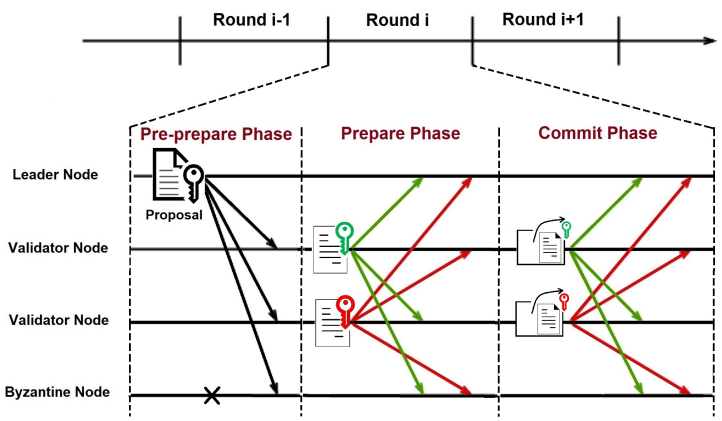

Figure 2: Overview of different phases in PBFT [19, 20, 21].

3 PBFT

PBFT [19, 20] was the first practical and efficient solution to deal with Byzantine faults in a weakly synchronous environment. It executes in some rounds between some nodes in a consensus group in which one node acts as the *leader*, and other nodes act as *validators*. As depicted in Figure 2, all messages are signed before broadcasting to other nodes in the consensus group. Each round consists of the following phases:

1. *Pre-prepare phase*: The leader initiates the consensus procedure by sending a signed pre-prepare message that is a block proposal containing a certain number of transactions.

2. *Prepare phase*: Upon receiving the pre-prepare message, each node in the consensus group checks the correctness and validity of the block and multicasts a signed prepare message (yes/no) to all other nodes.

3. *Commit phase*: Each node, after analysis of received prepare messages, multicasts a signed commit message (yes/no) to the consensus group. The block proposal is committed to the blockchain only if a sufficient number of nodes in the consensus group agree on it.

At the end of the above phases, all honest nodes in the consensus group reach a consensus about accepting or rejecting the block proposal and will have the same view about the blockchain state.

Since a consensus group of $\mathtt{t}$ players can tolerate $f = \lfloor(\mathtt{t}-1)/3\rfloor$ Byzantine nodes in the PBFT, each honest node needs to collect and verify at least $2f+1$ signatures in both prepare and commit phases, respectively. This might limit the application of PBFT to small consensus groups due to communication and computational overhead, but there are proposals for improving the scalability of PBFT and reducing its computational costs [21, 22, 38]. As it will be discussed in Section 6, recent proposals for BFT such as HotStuff [39] (and its variant LibraBFT [40]) are more efficient but new proposals would need more scrutiny [41] so we would prefer to deploy the well-studied PBFT in this paper.

4 Security Model

Based on basic functionalities that we consider for a PKI which includes registering, revoking, and updating public keys, an adversary might perform the following attacks:

- Registering a valid public key for an illegitimate entity

- Registering an invalid public key for an illegitimate entity
- Registering an invalid public key for a legitimate entity
- Updating public key of a legitimate entity with a valid/invalid public key
- Revoking public key of a legitimate entity

Validation of public keys is very important and invalid public keys can help an adversary to accomplish some attacks: In protocols based on the discrete logarithm problem, a small-subgroup attack might be feasible if public keys are not validated to be of prime order. A variant of this attack in elliptic curve-based schemes is the invalid-curve attack [42, 43]. We formalize such validations through the following definition.

Definition 7. We denote by *Public key and Identity Validation* (PIV) the procedure taken in the PKI for validation of public keys and identifiers:

$$\mathsf{PIV}(\mathsf{ID}, \mathsf{pk}) \rightarrow 1/\perp$$

We denote by $\mathbf{PK}$ and $\mathbf{ID}$ the list of all public keys and valid identifiers that are already registered in a PKI, respectively. We denote by $\boldsymbol{\mathcal{PK}}$ the space of all valid public keys generated by the $\mathsf{KG}(1^\lambda)$ algorithm using the randomness $r \in_R \{0,1\}^\lambda$:

$$\boldsymbol{\mathcal{PK}} = \{\mathsf{pk} | \exists r \in \{0,1\}^\lambda : \mathsf{KG}(1^\lambda; r) = (\mathsf{sk}, \mathsf{pk})\}$$

For a PKI that does not check the validity of public keys and identifiers, we have $\mathsf{PIV}(.,.) = 1$. A PKI may deploy zero-knowledge proofs through a so-called *proof-of-possession* (PoP) procedure to assure that any user owns the corresponding private key of its claimed public key. Although standards mandate the inclusion of PoP during registration, many existing PKIs do not require proofs of knowledge [44].

Remark 1. For a PKI that only considers *public key validation*, we define PIV as:

$$\mathsf{PIV}(\mathsf{ID}, \mathsf{pk}) = \begin{cases} 1 & \text{if } \mathsf{pk} \in \boldsymbol{\mathcal{PK}} \\ \perp & \text{otherwise} \end{cases}$$

Remark 2. For providing *public key uniqueness* in a PKI, we define PIV as:

$$\mathsf{PIV}(\mathsf{ID}, \mathsf{pk}) = \begin{cases} 1 & \text{if } (\mathsf{pk} \notin \mathbf{PK}) \wedge (\mathsf{pk} \in \boldsymbol{\mathcal{PK}}) \\ \perp & \text{otherwise} \end{cases}$$

Remark 3. For preserving *identity retention* (preventing different public keys registered for one identifier) in a PKI, we define PIV as:

$$\mathsf{PIV}(\mathsf{ID}, \mathsf{pk}) = \begin{cases} 1 & \text{if } (\mathsf{ID} \notin \mathbf{ID}) \wedge (\mathsf{pk} \in \boldsymbol{\mathcal{PK}}) \\ \perp & \text{otherwise} \end{cases}$$

We will use PIV to provide a generic model for PKI and to formalize the security definitions for an adversary that aims to register valid and invalid public keys for different identifiers of her will.

5 DBPKI

The proposed scheme DBPKI includes the following entities:

- *Root units* (R_i): Each root entity is part of the DBPKI and assumed to be honest in the beginning. It is identified by an identifier $\mathsf{ID}_{\mathrm{R}_i}$, and has a pair of private and public keys $(\mathsf{sk}_{\mathrm{R}_i}, \mathsf{pk}_{\mathrm{R}_i})$ and a fixed trust weight v_r. DBPKI is initialized by n root units $\{\mathrm{R}_1, \ldots, \mathrm{R}_n\}$.
- *Intermediate units* (I_i): Each intermediate entity is part of the DBPKI, identified by an identifier $\mathsf{ID}_{\mathrm{I}_i}$, and has a pair of private and public keys $(\mathsf{sk}_{\mathrm{I}_i}, \mathsf{pk}_{\mathrm{I}_i})$ and a fixed trust weight 1. Intermediate units could be organizations in applications like smart cities, and their number can be increased dynamically per se.
- *Ordinary units* (O_i): Each ordinary entity is *not* part of the DBPKI. It is identified by an identifier $\mathsf{ID}_{\mathrm{O}_i}$, and has a pair of private and public keys $(\mathsf{sk}_{\mathrm{O}_i}, \mathsf{pk}_{\mathrm{O}_i})$ and a fixed trust weight *zero*. This means that they cannot participate in enrollment and revocation procedures, but will be ordinary users of the system. They might be resource-constrained units that will join and use the system but will not be part of the DBPKI.

When referring to an entity in general, we denote it by u_i which belongs to one of above groups and has a $\texttt{role} \in \{\mathrm{R}, \mathrm{I}, \mathrm{O}\}$. Only root and intermediate units can participate in enrollment, revocation, and update procedures while any node can accomplish the verification procedure as described in Section 5.2. Any entity which is part the DBPKI, i.e. with $\texttt{role} \in \{\mathrm{R}, \mathrm{I}\}$, can initiate enrollment, revocation, or update procedures by establishing the PBFT consensus mechanism as a *leader* with $\mathsf{t} - 1$ number of other units that are part of the DBPKI. The block proposal will be added to the blockchain only if the consensus is achieved.

An *account* is identified by $(\mathsf{ID}_i, \mathsf{H}(\mathsf{pk}_{u_i}), \texttt{role})$ where ID_i is an identifier that uniquely identifies the account. Any entity u_i may have one or more accounts. By incorporating the public key validation function PIV into the DBPKI, only valid public keys will be accepted. This prevents further attacks and can support different policies such as public key uniqueness or identity retention.

5.1 Blockchain structure

Items that will be included in the blockchain are listed in Figure 3 where $\texttt{role} \in \{\mathrm{R}, \mathrm{I}, \mathrm{O}\}$ denotes the role of entity u_i with identifier ID_{u_i}, and $\texttt{flag} \in \{0, 1, 2\}$ denotes that public key pk_{u_i} is newly added, updated, or revoked, respectively. The very first block in the chain contains public keys and identifiers of all n root units: $\{(\mathsf{ID}_{\mathrm{R}_1}, \mathsf{pk}_{\mathrm{R}_1}, 0, \mathrm{R}), \ldots, (\mathsf{ID}_{\mathrm{R}_n}, \mathsf{pk}_{\mathrm{R}_n}, 0, \mathrm{R})\}$ together with a timestamp, block identifier, accumulator, and some additional data. The next blocks in the chain, as depicted in Figure 3, include a hash of public key instead of the public key itself. This will decrease the blockchain size. An arbitrary block BL_i may contain an arbitrary T number of key transaction items. Each valid block BL_i contains a secure one-way hash of the previous block in the chain $\mathsf{H}(\mathsf{BL}_{i-1})$ where i denotes the index of the current block. A single key transaction item includes adding, updating or revocation of public key of a unit u_i. A key transaction item is valid and will be added to the blockchain only after that a consensus is achieved between t number of units that are already part of the DBPKI. Any efficient and sound dynamic Merkle-tree based accumulator such as those proposed in [35, 36] can be used in the scheme.

5.2 Functionalities

The proposed scheme provides the following functionalities as defined in Figure 4:

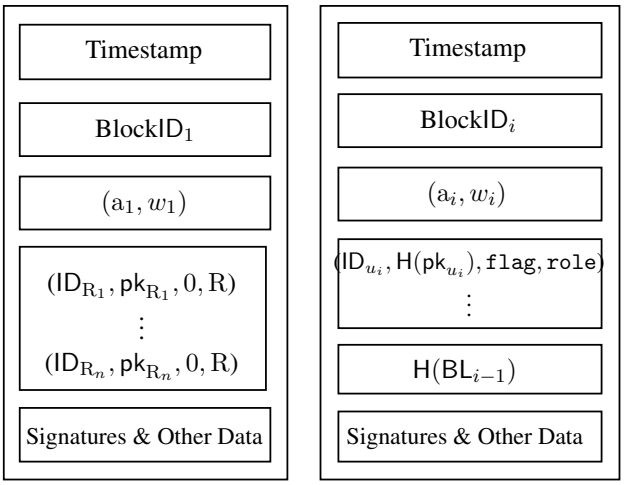

Figure 3: Contents of the first block BL_1 (left) and i^{th} block BL_i (right).

- $\mathsf{Enroll}(\mathsf{ID}_{u_i}, \mathsf{pk}_{u_i}, \mathtt{flag}, \mathtt{role}) \rightarrow 1/\perp$: Enrolls an entity with $\mathtt{role} \in \{\mathrm{R}, \mathrm{I}, \mathrm{O}\}$, identifier ID_{u_i}, and public key pk_{u_i}, and outputs success (1) or failure ($\perp$). Here, $\mathtt{flag}$ can be either 0 or 1 where 0 means pk_{u_i} has been registered for the first time for ID_{u_i}, and 1 means the public key has been updated due to an update procedure.
- $\mathsf{Revoke}(\mathsf{ID}_{u_i}, \mathsf{pk}_{u_i}) \rightarrow 1/\perp$: Revokes the public key pk_{u_i} corresponding to identity ID_{u_i}, and outputs success (1) or failure ($\perp$).
- $\mathsf{Update}(\mathsf{ID}_{u_i}, \mathsf{pk}^*_{u_i}) \rightarrow 1/\perp$: Updates the corresponding public key of an entity with identifier ID_{u_i} to $\mathsf{pk}^*_{u_i}$, and outputs success (1) or failure ($\perp$).
- $\mathsf{Verify}(\mathsf{ID}_{u_i}, \mathsf{pk}_{u_i}) \rightarrow 1/\perp$: Verifies whether or not pk_{u_i} is the corresponding public key of ID_{u_i}, and outputs success (1) or failure ($\perp$).

The enrollment, revocation, and update procedures require consensus between at least $\mathtt{t} - 1$ other units that are already part of the DBPKI where $\mathtt{t}$ is a system dependent parameter. To facilitate the election procedure within the consensus, $\mathtt{t}$ number of roots or intermediate nodes are assigned as members of the *consensus group*. The consensus group will be updated if some units are revoked at a later time or if the threshold value $\mathtt{t}$ is changed. PBFT is used as the consensus algorithm.

5.2.1 Initialization

1. Each of n root units $\{\mathrm{R}_1, \ldots, \mathrm{R}_n\}$ generates a key pair $(\mathsf{sk}_{\mathrm{R}_i}, \mathsf{pk}_{\mathrm{R}_i}) \leftarrow \mathsf{KG}(1^\lambda)$.
2. Unit R_1 creates the accumulator $\mathrm{a}_0 \leftarrow \mathsf{AccGen}(1^\lambda)$. It then adds identifiers and hash of public keys for all root units $(\mathsf{ID}_{\mathrm{R}_i}, \mathsf{H}(\mathsf{pk}_{\mathrm{R}_i}))$ to the accumulator where $i \in \{1, \ldots, n\}$. The accumulator and witnesses are denoted by a_1 and w_1 after those updates. It then generates a block proposal BL_1 which includes timestamp, BlockID$_1$, a_1, w_1, and $(\mathsf{ID}_{\mathrm{R}_i}, \mathsf{pk}_{\mathrm{R}_i}, 0, \mathrm{R})$ for $i \in \{1, \ldots, n\}$. Then, R_1 initiates the PBFT protocol as the leader and proceeds the consensus mechanism with the other root units.
3. All root units participate in the BPFT protocol and during the consensus procedure, verify that the accumulator is created correctly. At the end of the consensus procedure, they reach to agreement on BL_1 (and a_1). Otherwise, the initialization procedure starts from the beginning.

$\underline{\mathsf{Setup}(\lambda)}$
$\mathsf{TL} \leftarrow \emptyset$
$\mathrm{a}_0 \leftarrow \mathsf{AccGen}(1^\lambda)$
for $i \in \{1, \ldots, n\}$ **do**
  $(\mathsf{sk}_{u_i}, \mathsf{pk}_{u_i}) \leftarrow \mathsf{KG}(1^\lambda)$
  $\mathsf{TL} \xleftarrow{\cup} (\mathsf{ID}_{R_i}, \mathsf{pk}_{R_i}, 0, \mathrm{R})$
  $x \leftarrow (\mathsf{ID}_{R_i}, \mathsf{H}(\mathsf{pk}_{R_i}))$
  $\mathsf{AccAdd}(\mathrm{a}_0, x)$
$\mathsf{AddBL}(\{Opt, \mathrm{a}_1, w_1, \mathsf{TL}\})$

$\underline{\mathsf{AddBL}(Proposal)}$
if consensus achieved
  $\mathsf{BL}_{i+1} \xleftarrow{\cup} \{Proposal\}$

$\underline{\mathsf{Enroll}(\mathsf{ID}, \mathsf{pk}, \mathtt{flag}, \mathtt{role})}$
if $\mathsf{PIV}(\mathsf{ID}, \mathsf{pk}) = 1 \wedge$
$(|SI|_{valid} \geq 2f + 1)$ **then**
  $\mathsf{TL} \leftarrow (\mathsf{ID}, \mathsf{H}(\mathsf{pk}), 0, \mathtt{role})$
  $x \leftarrow (\mathsf{ID}, \mathsf{H}(\mathsf{pk}))$
  $(\mathrm{a}_{i+1}, w_{i+1}) \leftarrow \mathsf{AccAdd}(\mathrm{a}_i, x)$
  $\mathsf{AddBL}(\{Opt, \mathrm{a}_{i+1}, w_{i+1}, \mathsf{TL}\})$
  return 1
else
  return $\perp$

$\underline{\mathsf{Verify}(\mathsf{ID}, \mathsf{pk})}$
$x \leftarrow (\mathsf{ID}, \mathsf{H}(\mathsf{pk}))$
if $\mathsf{AccVer}(\mathrm{a}_i, x, w_i) = 1$
then return 1
else
  return $\perp$

$\underline{\mathsf{Revoke}(\mathsf{ID}, \mathsf{pk})}$
if $\mathsf{Verify}(\mathsf{ID}, \mathsf{pk}) \wedge$
$(|SI|_{valid} \geq 2f + 1)$ **then**
  $\mathsf{AccDel}(\mathrm{a}, (\mathsf{ID}, \mathsf{pk}))$
  $\mathsf{AddBL}(\mathsf{ID}, \mathsf{pk}, 2)$
else
  return $\perp$

$\underline{\mathsf{Update}(\mathsf{ID}, \mathsf{pk}^{old}, \mathsf{pk}^{new})}$
if $\mathsf{PIV}(\mathsf{ID}, \mathsf{pk}^{new}) = 1$**then**
  $\mathsf{Revoke}(\mathsf{ID}, \mathsf{pk}^{old})$
  $\mathsf{Enroll}(\mathsf{ID}, \mathsf{pk}^{new}, 1, \mathtt{role})$
  return 1
else
  return $\perp$

Figure 4: Definitions for basic functionalities in DBPKI.

5.2.2 Enrollment procedure

Any arbitrary unit u_i that wants to join the DBPKI needs to generate a private-public key pair (sk_{u_i}, pk_{u_i}). Then, u_i can initiate the enrollment procedure by sending a enrollment request to another unit u_j which is in the consensus group. The enrollment request includes a data structure that we refer to as *Enrollment Proof (*EP*)*, and a *public key item* (PI). EP indicates specifications of the enrollment (and further information when the unit has already a certificate from another trusted PKI as will be discussed in Section 6). PI includes $\{\mathsf{ID}_{u_i}, \mathsf{H}(\mathsf{pk}_{u_i}), \mathtt{flag}, \mathtt{role}, \mathsf{Opt}\}$ in which Opt denotes additional data such as requested validity period of the key if applicable, subject information, etc. Then, any unit u_j in the consensus group verifies the claimed public key and checks if u_i is allowed to join the DBPKI. Then, u_j signs the proposal with its own private key sk_{u_j} and proceeds with the PBFT consensus mechanism with the other units in the consensus group. The block proposal will be added to the ledger only after a successful consensus. Steps for the enrollment procedure can be followed as:

1. u_i generates a key pair $(\mathsf{sk}_{u_i}, \mathsf{pk}_{u_i}) \leftarrow \mathsf{KG}(1^\lambda)$. It then generates an enrollment proof EP, a *public key item* $\mathsf{PI} \leftarrow \{\mathsf{ID}_{u_i}, \mathsf{H}(\mathsf{pk}_{u_i}), \mathtt{flag}, \mathtt{role}, \mathsf{Opt}\}$, a *signature item* $\mathsf{SI} \leftarrow \mathsf{Sign}(\mathsf{sk}_{u_i}, \mathsf{PI})$, and an *enrollment request* which contains $\{\mathsf{EP}, \mathsf{PI}, \mathsf{SI}, \mathsf{pk}_{u_i}\}$ and will be sent to the consensus group.

2. Any unit u_j in the consensus group that has received an enrollment request $\{\mathsf{EP}, \mathsf{PI}, \mathsf{SI}, \mathsf{pk}_{u_i}\}$ and trusts u_i, checks that `flag` and `role` have correct values and verifies the signature included in the enrollment request. If $\mathsf{Veri}(\mathsf{sk}_{u_i}, \mathsf{SI}, \mathsf{PI}) = 1$ then u_j verifies that the public key pk_{u_i} is valid: If

$\mathsf{PIV}(ID_{u_i}, \mathsf{pk}_{u_i}) = 1$ then u_j signs the received PI using its private key sk_{u_j}, generates another *signature item* $\mathsf{SI} \leftarrow \mathsf{Sign}(\mathsf{sk}_{u_j}, \mathsf{PI})$, broadcasts $\{\mathsf{PI}, \mathsf{SI}\}$ to the other units in the consensus group, and proceeds according to steps in the PBFT consensus mechanism.

3. During the consensus procedure, any unit in the consensus group verifies that any signature item SI from other units is valid and its signer is part of the DBPKI. If any signature is not verified, the corresponding pair of $\{\mathsf{PI}, \mathsf{SI}\}$ will be discarded. Upon successful completion of the consensus mechanism in previous step, which means there are at least $2\lfloor(\mathsf{t}-1)/3\rfloor + 1$ number of valid $\{\mathsf{PI}, \mathsf{SI}\}$ for a particular PI, all the honest units in the consensus group accept the proposal for enrollment of u_i and have the same view about the blockchain state. All the units in the consensus group also add $x = (\mathsf{ID}_{u_i}, \mathsf{H}(\mathsf{pk}_{u_i}))$ to the accumulator: $(\mathrm{a}_{i+1}, w^x_{i+1}, \mathsf{updmsg}_{i+1}) \leftarrow \mathsf{AccAdd}(\mathrm{a}_i, x)$. The new block BL_m that will be added to the ledger contains a key transaction item TI which includes PI and all corresponding SIs during the consensus procedure together with the updated accumulator and witness. BL_m will be broadcast to the whole network.

4. All block recipients verify that the accumulator has been updated correctly. If verified, they update their stored accumulator values with a, and update their stored witnesses as $w^x_{i+1} \leftarrow \mathsf{AccWitAdd}(w^x_i, y, \mathsf{updmsg}_{i+1})$ where y denotes the newly added pairs of identifiers and public keys. Otherwise, they discard the block or initiate the revocation procedure.

5.2.3 Revocation procedure

Any arbitrary entity u_i which is part of the DBPKI can initiate the revocation procedure. The revocation procedure starts when u_i finds a reason to revoke an existing key of entity u_t in the ledger. This can happen due to bad or malicious behavior, due to that the key is obsolete, upon request from the owner of a key when its private key has been compromised, or other reasons. Entity u_i sends a revocation request, and broadcasts it to the consensus group. The revocation request includes a data structure that we refer to it as *Revocation Proof (*RP*)* which indicates a reason for the revocation request. Any unit in the consensus group, checks RP before further processing of the revocation request. Steps for the revocation procedure are as follows:

1. u_i generates a revocation proof RP, a *public key item* (PI) as $\mathsf{PI} \leftarrow \{\mathsf{ID}_{u_i}, \mathsf{H}(\mathsf{pk}_{u_i}), 2, \texttt{role}, \mathsf{Opt}\}$ where 2 denotes the `flag` for revocation, and a signature item $\mathsf{SI} \leftarrow \mathsf{Sign}(\mathsf{sk}_{u_i}, \mathsf{PI})$. It then sends the revocation request $\{\mathsf{RP}, \mathsf{PI}, \mathsf{SI}, \mathsf{pk}_{u_i}\}$ to the consensus group which includes at least $\mathsf{t}-1$ other entities that are already part of the DBPKI and might accept revoking the public key of u_t.

2. Any node u_j in the consensus group that has received the revocation request checks the received RP and verifies the signature item SI. If both are verified, u_j signs the received PI with its private key sk_{u_j} and generates a signature item $\mathsf{SI} \leftarrow \mathsf{Sign}(\mathsf{sk}_{u_j}, \mathsf{PI})$, and broadcasts $\{\mathsf{PI}, \mathsf{SI}\}$ to other units in the consensus group, and proceeds according to steps in the PBFT consensus mechanism.

3. During the consensus procedure, any unit in the consensus group verifies that any signature item SI from other units is valid and its signer is part of the DBPKI. If any signature is not verified, the corresponding pair of $\{\mathsf{PI}, \mathsf{SI}\}$ will be excluded from further consideration. Upon successful completion of the consensus mechanism in previous step which means there are at least $2\lfloor(\mathsf{t}-1)/3\rfloor + 1$ number of valid $\{\mathsf{PI}, \mathsf{SI}\}$ for a particular PI, all the honest units in the consensus group accept the proposal for revoking u_i and have the same view about the blockchain state. All the units in the consensus group also include PI and all valid SIs together with identifiers and hash of public keys of the signing entities into a new key transaction item TI. They also verify that $x = (\mathsf{ID}_{u_t}, \mathsf{H}(\mathsf{pk}_{u_t}))$ exists in the accumulator by checking that $\mathsf{AccVer}(\mathrm{a}_i, x, w^x_i) = 1$. If the verification fails, the procedure aborts. Otherwise, they delete $(\mathsf{ID}_{u_t}, \mathsf{H}(\mathsf{pk}_{u_t}))$ from the accumulator by executing $(\mathrm{a}_{i+1}, \mathsf{updmsg}_{i+1}) \leftarrow \mathsf{AccDel}(\mathrm{a}_i, x)$,

and append the updated accumulator a_{i+1} to TI. The new block BL_m that will be added to the ledger contains a key transaction item TI which includes PI and all corresponding SIs during the consensus procedure, together with the updated accumulator and witnesses. BL_m will be broadcast to the whole network.

4. All block recipients verify that the accumulator has been updated correctly. If verified, they update their stored accumulator values with a, and update their stored witnesses as $w^x_{i+1} \leftarrow \mathsf{MemWitUpdateDel}(x, w^x_i, \mathsf{updmsg}_{i+1})$. Otherwise, they discard the block.

5.2.4 Update procedure

The update procedure should naturally include revoking the old key $\mathsf{pk}^{old}_{u_i}$ by executing $\mathsf{Revoke}(\mathsf{ID}_{u_i}, \mathsf{pk}^{old}_{u_i})$ and registering the new key $\mathsf{pk}^{new}_{u_i}$ by executing $\mathsf{Enroll}(\mathsf{ID}_{u_i}, \mathsf{pk}^{new}_{u_i}, 1, \texttt{role})$ where $\texttt{flag} = 1$ means that there are previous records for the public key in the ledger and it has been updated. Since Enroll and Revoke procedures require consensus between $\mathtt{t}$ nodes, for the sake of efficiency, both procedures will be merged, i.e. the block proposal in the consensus mechanism includes records for revoking $\mathsf{pk}^{old}_{u_i}$ and enrollment of $\mathsf{pk}^{new}_{u_i}$. The new block that will be added to the ledger contains $(\mathsf{ID}_{u_i}, \mathsf{H}(\mathsf{pk}^{old}_{u_i}), 2, \texttt{role})$ and $(\mathsf{ID}_{u_i}, \mathsf{H}(\mathsf{pk}^{new}_{u_i}), 1, \texttt{role})$.

5.2.5 Verification procedure

The verification procedure is essentially a proof of membership algorithm and is used to verify whether or not a given public key belongs to a given identifier. It naturally considers the revocation status of a key and will respond with true if and only if the public key is valid and belongs to the given identifier. This can decrease the computational costs, especially for resource-constrained environments. For a given $x = (\mathsf{ID}_{u_j}, \mathsf{H}(\mathsf{pk}_{u_j}))$, any node can accomplish the verification procedure by verifying that $\mathsf{AccVer}(\mathrm{a}_i, x, w^x) = 1$.

6 Further discussion

The computational and communication costs of the DBPKI depend mostly on the deployed dynamic accumulator and consensus mechanism. Table 2 shows the computational costs of different procedures in the DBPKI. Computational costs of different accumulator's algorithms in some constructions are shown in Table 3. Accumulators with low-frequency updates require fewer updates which decreases the communications overhead. The consensus procedure is one of the costly parts and its costs depend on the number of nodes in the consensus group $\mathtt{t}$ and the deployed algorithm. There is a trade-off between security and efficiency. Increasing $\mathtt{t}$ will naturally increase security (since to withstand against f corrupt nodes, we need to have at least $\mathtt{t} = 3f + 1$ nodes in the consensus group) but decreases efficiency (as computational costs and communications overhead will increase). PBFT is not the most efficient BFT algorithm but is well-studied. PBFT is safe against f Byzantine faults, safe against asynchrony, and responsive (i.e., a leader node can propose without delay). However, the number of messages for a consensus decision and the number of messages to rotate a leader are both quadratic. There were also other options that we could deploy: Mir-BFT [38] is a recent proposal based on the PBFT that allows having multiple leaders instead of one leader and improves the throughput by requiring fewer signatures. HotStuff [39] and its underdeveloped variant, LibraBFT [40], are other proposals that claim to provide the same properties as the PBFT but with a linear number of messages for a consensus decision and rotating a leader. We could incorporate those new BFT protocols into the scheme to make it more efficient and scalable but we simply used a classical BFT in this

Procedure	Sign/Veri	AccAdd	AccWitAdd*	MemWitUpdateDel*	AccDel	AccVer
Enroll	$O(\mathsf{t}^2)$	t	k	-	-	-
Revoke	$O(\mathsf{t}^2)$	-	-	k	t	t
Update	$O(\mathsf{t}^2)$	t	k	k	t	t
Verify	-	-	-	-	-	1

Table 2: Total computational costs of different procedures in DBPKI where t denotes the number of nodes in the consensus group, and k denotes the number of nodes that store and validate the accumulator and witnesses.

Scheme	Sign	Merkle	BraavosB [36]	CL-RSA-B [36]	Braavos [36]
AccAdd	1	$\log d$	1	1	1
AccWitAdd	0	$\log d$	0	0	0
MemWitUpdateDel	0	$\log d$	1	1	1
AccDel	0	$\log d$	1	1	1
AccVer	1	$\log d$	1	1	1

Table 3: Costs of incorporating different accumulators into procedures where d denotes the number of elements added to the accumulator.

paper since new proposals would require further scrutiny and some of them have security vulnerabilities [41].

Other alternatives to the proposed scheme could also be considered: A variant that could reduce the number of required signatures during the enrollment and revocation procedures is to incorporate the trust weights in reaching the threshold. Since the trust weight of root and intermediate units is defined as w_r and 1, respectively, each signature from a root entity could worth like w_r signatures from intermediate entities. This reduces the waiting time which could be useful in emergency cases but it requires some changes in the consensus mechanism.

Another concern regarding the proposed solution is the interoperability with the existing CA-based PKI. This could be done by establishing a trust policy where members of the consensus group would consider certificates issued by some CAs as trusted. Then, any unit having a certificate from a trusted CA can include such specification in the enrollment proof EP that is generated through the enrollment procedure. An example case would be a vehicle holding a certificate from a CA in another country that wants to visit a smart city deploying the DBPKI. However, this should be a temporary enrollment and the unit shall join as an ordinary unit especially if the other CA-based PKI does not provide certificate transparency.

In the rest of this section, we briefly argue about the security of the DBPKI. Our security definitions follow a provable security game-based approach. For the security model defined in Section 4 and basic functionalities of the DBPKI defined in Figure 4, we can define different security experiments as depicted in Figure 5. The security of the DBPKI stems from the following assumptions: We assume that all deployed hash functions are secure according to definition 2, all digital signatures are EUF-CMA-secure according to definition 3, and the deployed dynamic accumulator is correct and sound according to definition 5. The security of enrollment, revocation, and update procedures stems from the PBFT consensus mechanism and

$$\underline{\mathbf{Exp}^{\mathsf{Enroll\text{-}ValidKey\text{-}IllegitimateEntity}}(\mathcal{A})}$$
$$(\mathsf{ID}, \mathsf{pk}) \xleftarrow{\$} \mathcal{A}()$$
$$\textbf{return } 1 \textbf{ if } (\mathsf{Enroll}(\mathsf{ID}, \mathsf{pk}, \texttt{flag}, \texttt{role}) = 1) \wedge (\mathsf{Verify}(\mathsf{ID}, \mathsf{pk}) = \perp) \wedge (\mathsf{PIV}(\mathsf{ID}, \mathsf{pk}) = 1)$$

$$\underline{\mathbf{Exp}^{\mathsf{Enroll\text{-}InvalidKey\text{-}IllegitimateEntity}}(\mathcal{A})}$$
$$(\mathsf{ID}, \mathsf{pk}) \xleftarrow{\$} \mathcal{A}()$$
$$\textbf{return } 1 \textbf{ if } (\mathsf{Enroll}(\mathsf{ID}, \mathsf{pk}, \texttt{flag}, \texttt{role}) = 1) \wedge (\mathsf{Verify}(\mathsf{ID}, \mathsf{pk}) = \perp) \wedge (\mathsf{PIV}(\mathsf{ID}, \mathsf{pk}) = \perp)$$

$$\underline{\mathbf{Exp}^{\mathsf{Enroll\text{-}InvalidKey\text{-}LegitimateEntity}}(\mathcal{A})}$$
$$(\mathsf{ID}, \mathsf{pk}) \xleftarrow{\$} \mathcal{A}()$$
$$\textbf{return } 1 \textbf{ if } (\mathsf{Enroll}(\mathsf{ID}, \mathsf{pk}, \texttt{flag}, \texttt{role}) = 1) \wedge (\mathsf{Verify}(\mathsf{ID}, \mathsf{pk}) = 1) \wedge (\mathsf{PIV}(\mathsf{ID}, \mathsf{pk}) = \perp)$$

$$\underline{\mathbf{Exp}^{\mathsf{Update\text{-}Collision}}(\mathcal{A})}$$
$$(\mathsf{ID}, \mathsf{pk}^*) \xleftarrow{\$} \mathcal{A}()$$
$$\textbf{return } 1 \textbf{ if } (\mathsf{Update}(\mathsf{ID}, \mathsf{pk}^{old}, \mathsf{pk}^*) = 1) \wedge (\mathsf{Verify}(\mathsf{ID}, \mathsf{pk}^*) = 1)$$

$$\underline{\mathbf{Exp}^{\mathsf{Revoke\text{-}Collision}}(\mathcal{A})}$$
$$(\mathsf{ID}, \mathsf{pk}) \xleftarrow{\$} \mathcal{A}()$$
$$\textbf{return } 1 \textbf{ if } (\mathsf{Revoke}(\mathsf{ID}, \mathsf{pk}) = 1) \wedge (\mathsf{Verify}(\mathsf{ID}, \mathsf{pk}) = \perp)$$

Figure 5: Security definitions for the DBPKI.

the deployed accumulator. Hence, we can discuss how many honest nodes would be required to guarantee security.

Let $\mathtt{t}$ denote the number of nodes in the consensus group where up to f players are corrupted, and $h = t - f$ denotes the minimal number of honest players. For unconditionally secure protocols in the standard model with a complete and synchronous network of bilateral authenticated communication channels among the players and no trusted entity, a resilient broadcast is achievable if and only if $2\mathtt{t}/3 < h$ [45]. PBFT withstands against at most $f = \lfloor(\mathtt{t} - 1)/3\rfloor$ Byzantine nodes. Correctness proofs can be found in [46]. We assume that the network is controlled by an adversary that can reorder or delay messages under synchronous compliance and thus the protocol does not deviate from the synchrony assumption.

Theorem 1. If the deployed hash function is collision-resistant, the deployed signature schemes are EUF-CMA-secure, the deployed accumulator is sound, and not more than $f = \lfloor(\mathtt{t} - 1)/3\rfloor$ nodes in the consensus group are Byzantine, then an adversary $\mathcal{A}$ has a negligible advantage in winning the security experiments described in Figure 5.

The theorem can be proved using sequences of games. In the first experiment $\mathbf{Exp}^{\mathsf{Enroll\text{-}ValidKey\text{-}IllegitimateEntity}}$, an adversary $\mathcal{A}$ wins if $\mathcal{A}$ can register a valid public key for an illegitimate entity. In the second experiment $\mathbf{Exp}^{\mathsf{Enroll\text{-}InvalidKey\text{-}IllegitimateEntity}}$, $\mathcal{A}$ wins if $\mathcal{A}$ can register an invalid public key for an illegitimate entity. $\mathcal{A}$ wins the third experiment

$\mathbf{Exp}^{\mathsf{Enroll\text{-}InvalidKey\text{-}LegitimateEntity}}$ if $\mathcal{A}$ can register an invalid public key for a legitimate entity. In the fourth experiment $\mathbf{Exp}^{\mathsf{Update\text{-}Collision}}$, $\mathcal{A}$ wins if $\mathcal{A}$ can update/change the public key of a legitimate and enrolled entity with another public key of its choice. $\mathcal{A}$ wins the fifth experiment $\mathbf{Exp}^{\mathsf{Revoke\text{-}Collision}}$ if $\mathcal{A}$ can revoke the public key of a legitimate and enrolled entity without its consent. Note that the security of the first three experiments that concern the enrollment is equivalent, i.e. if $\mathcal{A}$ wins one of them, $\mathcal{A}$ will win the two others.

7 Conclusion

A decentralized blockchain-based PKI (DBPKI) was introduced in this paper that eliminates the single-point-of-failure and demanding needs for maintaining CA and revocation lists in the traditional PKIs. It distributes trust between entities through WoT and new keys are enrolled or revoked based on a consensus mechanism between trusted nodes that are already part of the system. Although we used PBFT, it can be replaced with more scalable and efficient variants. For efficient verification of public keys, a dynamic cryptographic accumulator is incorporated into the scheme that makes it suitable for IoT applications. DBPKI provides different functionalities for registration, update, verification, and revocation of keys. All operations are transparent and auditable.

Acknowledgment

This research was supported by the Swedish Foundation for Strategic Research under Grant No. RIT-0035.

References

[1] M. Y. Kubilay, M. S. Kiraz, and H. A. Mantar, "Certledger: A new PKI model with certificate transparency based on blockchain," *Comput. Secur.*, vol. 85, pp. 333–352, 2019. [Online]. Available: https://doi.org/10.1016/j.cose.2019.05.013

[2] B. Laurie, A. Langley, and E. Kasper, "RFC 6962: Certificate transparency," *Request for Comments. IETF*, 2013.

[3] M. D. Ryan, "Enhanced certificate transparency and end-to-end encrypted mail," in *21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, February 23-26, 2014*, 2014. [Online]. Available: https://www.ndss-symposium.org/ndss2014/enhanced-certificate-transparency-and-end-end-encrypted-mail

[4] D. A. Basin, C. J. F. Cremers, T. H. Kim, A. Perrig, R. Sasse, and P. Szalachowski, "ARPKI: attack resilient public-key infrastructure," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, 2014, pp. 382–393. [Online]. Available: https://doi.org/10.1145/2660267.2660298

[5] P. Szalachowski, S. Matsumoto, and A. Perrig, "PoliCert: Secure and flexible TLS certificate management," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, 2014, pp. 406–417. [Online]. Available: https://doi.org/10.1145/2660267.2660355

[6] M. S. Melara, A. Blankstein, J. Bonneau, E. W. Felten, and M. J. Freedman, "CONIKS: bringing key transparency to end users," in *24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015*, 2015, pp. 383–398. [Online]. Available: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/melara

[7] J. Bonneau, "EthIKS: Using ethereum to audit a CONIKS key transparency log," in *Financial Cryptography and Data Security - FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers*, ser. Lecture Notes in Computer Science, J. Clark, S. Meiklejohn, P. Y. A. Ryan, D. S. Wallach, M. Brenner, and K. Rohloff, Eds., vol. 9604. Springer, 2016, pp. 95–105. [Online]. Available: https://doi.org/10.1007/978-3-662-53357-4_7

[8] J. Yu, V. Cheval, and M. Ryan, "DTKI: A New Formalized PKI with Verifiable Trusted Parties," *The Computer Journal*, vol. 59, no. 11, pp. 1695–1713, 11 2016. [Online]. Available: https://doi.org/10.1093/comjnl/bxw039

[9] T. H. Kim, L. Huang, A. Perrig, C. Jackson, and V. D. Gligor, "Accountable key infrastructure (AKI): a proposal for a public-key validation infrastructure," in *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*, 2013, pp. 679–690. [Online]. Available: https://doi.org/10.1145/2488388.2488448

[10] Z. Wang, J. Lin, Q. Cai, Q. Wang, J. Jing, and D. Zha, "Blockchain-based certificate transparency and revocation transparency," in *Financial Cryptography and Data Security - FC 2018 International Workshops, BITCOIN, VOTING, and WTSC, Nieuwpoort, Curaçao, March 2, 2018, Revised Selected Papers*, 2018, pp. 144–162. [Online]. Available: https://doi.org/10.1007/978-3-662-58820-8_11

[11] K. Lewison and F. Corella, "Backing rich credentials with a blockchain PKI," *Tech. Rep.*, 2016.

[12] K. Yang, J. J. Sunny, and L. Wang, "Blockchain-based decentralized public key management for named data networking," in *The International Conference on Computer Communications and Networks (ICCCN 2018)*, 2018.

[13] B. Khieu and M. Moh, "CBPKI: cloud blockchain-based public key infrastructure," in *Proceedings of the 2019 ACM Southeast Conference, ACM SE '19, Kennesaw, GA, USA, April 18-20, 2019*, D. Lo, D. Kim, and E. Gamess, Eds. ACM, 2019, pp. 58–63. [Online]. Available: https://doi.org/10.1145/3299815.3314433

[14] S. Matsumoto and R. M. Reischuk, "IKP: turning a PKI around with blockchains," *IACR Cryptol. ePrint Arch.*, vol. 2016, p. 1018, 2016. [Online]. Available: http://eprint.iacr.org/2016/1018

[15] M. Ali, J. C. Nelson, R. Shea, and M. J. Freedman, "Blockstack: A global naming and storage system secured by blockchains," in *2016 USENIX Annual Technical Conference, USENIX ATC 2016, Denver, CO, USA, June 22-24, 2016*, 2016, pp. 181–194. [Online]. Available: https://www.usenix.org/conference/atc16/technical-sessions/presentation/ali

[16] M. Al-Bassam, "SCPKI: A smart contract-based PKI and identity system," in *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*, ser. BCC '17. New York, NY, USA: ACM, 2017, pp. 35–40. [Online]. Available: http://doi.acm.org/10.1145/3055518.3055530

[17] B. Qin, J. Huang, Q. Wang, X. Luo, B. Liang, and W. Shi, "Cecoin: A decentralized PKI mitigating MitM attacks," *Future Gener. Comput. Syst.*, vol. 107, pp. 805–815, 2020. [Online]. Available: https://doi.org/10.1016/j.future.2017.08.025

[18] C. Fromknecht, D. Velicanu, and S. Yakoubov, "A decentralized public key infrastructure with identity retention," *IACR Cryptology ePrint Archive*, vol. 2014, p. 803, 2014. [Online]. Available: http://eprint.iacr.org/2014/803

[19] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *Proceedings of the Third USENIX Symposium on Operating Systems Design and Implementation (OSDI), New Orleans, Louisiana, USA, February 22-25, 1999*, M. I. Seltzer and P. J. Leach, Eds. USENIX Association, 1999, pp. 173–186. [Online]. Available: https://dl.acm.org/citation.cfm?id=296824

[20] ——, "Practical Byzantine fault tolerance and proactive recovery," *ACM Trans. Comput. Syst.*, vol. 20, no. 4, pp. 398–461, 2002. [Online]. Available: https://doi.org/10.1145/571637.571640

[21] X. Fan, "Scalable practical Byzantine fault tolerance with short-lived signature schemes," in *Proceedings of the 28th Annual International Conference on Computer Science and Software Engineering, CASCON 2018, Markham, Ontario, Canada, October 29-31, 2018*, I. Onut, A. Jaramillo, G. Jourdan, D. C. Petriu, and W. Chen, Eds. ACM, 2018, pp. 245–256. [Online]. Available: https://dl.acm.org/citation.cfm?id=3291316

[22] H. Xu, Y. Long, Z. Liu, Z. Liu, and D. Gu, "Dynamic practical Byzantine fault tolerance," in *2018 IEEE Conference on Communications and Network Security, CNS 2018, Beijing, China, May 30 - June 1, 2018*. IEEE, 2018, pp. 1–8. [Online]. Available: https://doi.org/10.1109/CNS.2018.8433150

[23] S. Santesson, M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams, "X.509 internet public key infrastructure online certificate status protocol - OCSP," *RFC*, vol. 6960, pp. 1–41, 2013. [Online]. Available: https://doi.org/10.17487/RFC6960

[24] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. T. Polk, "Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile," *RFC*, vol. 5280, pp. 1–151, 2008. [Online]. Available: https://doi.org/10.17487/RFC5280

[25] J. Benaloh and M. de Mare, "One-way accumulators: A decentralized alternative to digital signatures," in *Workshop on the Theory and Application of Cryptographic Techniques on Advances in Cryptology*, ser. EUROCRYPT '93. Berlin, Heidelberg: Springer-Verlag, 1994, pp. 274–285.

[26] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[27] C. Shen and F. Peña-Mora, "Blockchain for cities - A systematic literature review," *IEEE Access*, vol. 6, pp. 76 787–76 819, 2018. [Online]. Available: https://doi.org/10.1109/ACCESS.2018.2880744

[28] 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

[29] L. M. Bach, B. Mihaljevic, and M. Zagar, "Comparative analysis of blockchain consensus algorithms," in *41st International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2018, Opatija, Croatia, May 21-25, 2018*, 2018, pp. 1545–1550. [Online]. Available: https://doi.org/10.23919/MIPRO.2018.8400278

[30] H. Anada, J. Kawamoto, J. Weng, and K. Sakurai, "Identity-embedding method for decentralized public-key infrastructure," in *Trusted Systems - 6th International Conference, INTRUST 2014, Beijing, China, December 16-17, 2014, Revised Selected Papers*, 2014, pp. 1–14. [Online]. Available: https://doi.org/10.1007/978-3-319-27998-5_1

[31] 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

[32] 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

[33] O. Omolola and P. Plessing, "Revisiting privacy-aware blockchain public key infrastructure," *IACR Cryptology ePrint Archive*, vol. 2019, p. 527, 2019. [Online]. Available: https://eprint.iacr.org/2019/527

[34] S. Goldwasser, S. Micali, and R. L. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," *SIAM J. Comput.*, vol. 17, no. 2, pp. 281–308, 1988. [Online]. Available: https://doi.org/10.1137/0217017

[35] L. Reyzin and S. Yakoubov, "Efficient asynchronous accumulators for distributed PKI," in *Security and Cryptography for Networks - 10th International Conference, SCN 2016, Amalfi, Italy, August 31 - September 2, 2016, Proceedings*, 2016, pp. 292–309. [Online]. Available: https://doi.org/10.1007/978-3-319-44618-9_16

[36] F. Baldimtsi, J. Camenisch, M. Dubovitskaya, A. Lysyanskaya, L. Reyzin, K. Samelin, and S. Yakoubov, "Accumulators with applications to anonymity-preserving revocation," in *2017 IEEE European Symposium on Security and Privacy, EuroS&P 2017, Paris, France, April 26-28, 2017*, 2017, pp. 301–315. [Online]. Available: https://doi.org/10.1109/EuroSP.2017.13

[37] J. Camenisch and A. Lysyanskaya, "Dynamic accumulators and application to efficient revocation of anonymous credentials," in *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, 2002, pp. 61–76. [Online]. Available: https://doi.org/10.1007/3-540-45708-9_5

[38] C. Stathakopoulou, T. David, and M. Vukolic, "Mir-BFT: High-throughput BFT for blockchains," *CoRR*, vol. abs/1906.05552, 2019. [Online]. Available: http://arxiv.org/abs/1906.05552

[39] M. Yin, D. Malkhi, M. K. Reiter, G. Golan-Gueta, and I. Abraham, "HotStuff: BFT consensus with linearity and responsiveness," in *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*, P. Robinson and F. Ellen, Eds. ACM, 2019, pp. 347–356. [Online]. Available: https://doi.org/10.1145/3293611.3331591

[40] LibraBFT Team,, "State machine replication in the Libra blockchain," 2020. [Online]. Available: https://developers.libra.org/docs/assets/papers/libra-consensus-state-machine-replication-in-the-libra-blockchain/2020-05-26.pdf

[41] A. Momose and J. P. Cruz, "Force-locking attack on Sync Hotstuff," *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 1484, 2019. [Online]. Available: https://eprint.iacr.org/2019/1484

[42] A. Antipa, D. R. L. Brown, A. Menezes, R. Struik, and S. A. Vanstone, "Validation of elliptic curve public keys," in *Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA, January 6-8, 2003, Proceedings*, 2003, pp. 211–223. [Online]. Available: https://doi.org/10.1007/3-540-36288-6_16

[43] M. Toorani, "Security analysis of the IEEE 802.15.6 standard," *Int. J. Commun. Syst.*, vol. 29, no. 17, pp. 2471–2489, 2016. [Online]. Available: https://doi.org/10.1002/dac.3120

[44] T. Ristenpart and S. Yilek, "The power of proofs-of-possession: Securing multiparty signatures against rogue-key attacks," in *Advances in Cryptology - EUROCRYPT 2007, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007, Proceedings*, 2007, pp. 228–245. [Online]. Available: https://doi.org/10.1007/978-3-540-72540-4_13

[45] J. Considine, M. Fitzi, M. K. Franklin, L. A. Levin, U. M. Maurer, and D. Metcalf, "Byzantine agreement given partial broadcast," *J. Cryptology*, vol. 18, no. 3, pp. 191–217, 2005. [Online]. Available: https://doi.org/10.1007/s00145-005-0308-x

[46] M. Castro, B. Liskov *et al.*, "A correctness proof for a practical Byzantine-fault-tolerant replication algorithm," Technical Memo MIT/LCS/TM-590, MIT Laboratory for Computer Science, Tech. Rep., 1999.