AcBF: A Revocable Blockchain-based Identity Management Enabling Low-Latency Authentication

Jianan Hong*§, Jiayue Zhou†, Yuqing Li‡, Jia Cheng*, Cunqing Hua*
*School of Cyber Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China
†SJTU-ParisTech Elite Institute of Technology, Shanghai Jiao Tong University, Shanghai 200240, China
‡School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China
§Corresponding author: hongjn@sjtu.edu.cn

***Abstract*—Blockchain-based identity brings in great evolution due to its decentralized deployment, transparent and tamper-free ledger. Specification groups of B5G/6G are exploring into integrate the technology to future network systems, e.g., Internet of Things, vehicular network, industrial communications. However, devices in these systems often have storage constraints and unstable channels, which necessitates lightweight node deployment. The security issue arises: revoked identity can forge a legitimate authentication, since the lightweight verifier does not maintain the revocation transactions. This paper hence proposes AcBF, a novel revocable identity management scheme, that enables extremely low authentication latency by allowing the lightweight node to query the certificate's status locally. To realize this feature trustfully, we design a revocation transaction based on accumulator-assisted Bloom filter to minimize the storage of certificate status structure. Secondly, we construct the blockchain protocol to ensure that no revocation event slips on any lightweight ledger, even in an insecure or unstable communication environment. In addition, different from other revocation mechanisms, AcBF minimizes the impact on valid users during the revocation process. Through security and performance analysis, AcBF has shown strong security and advantageous efficiency on both lightweight verifiers and certificate owners, thus suits identity management systems with low-latency constraints.**

***Index Terms*—Blockchain identity, Revocation-aware authentication, Lightweight node, Low latency**

I. Introduction

Blockchain has become increasingly popular in distributed identity management [1] due to its security features, such as append-only and non-tempering storage, decentralized consensus, and reliable trust establishment. These features make blockchain-based systems particularly well-suited for managing trust across a large number of entities [2], [3], compared to traditional identity systems like Public Key Infrastructure (PKI) [4]. Therefore, the idea of blockchain-based identity can greatly enhance the security of various network scenarios, including Internet of Thing (IoT) [5], [6] and Internet of Vehicles (IoV) [7], [8], where decentralized trust management is crucial.

Despite the advantages, the decentralized trust organization introduced by blockchain also brings in many practical problems because of its certificate status query method. Typically, in a blockchain-based identity system, a certificate's status (e.g., registered, illegal, or revoked) should be queried from blockchain ledger, which is maintained by multiple nodes. Such practice poses unique challenges to communication resource-constrained scenarios. Firstly, the transaction query requires several round trip time (RTT) between IoT devices (or vehicle) and at least one blockchain node, which leads to unbearable delay for latency-strict cases, e.g., safety notification message of IoV. Additionally, the highly dynamic topology of these network system makes the routing of query response complex. Secondly, querying certificate status from just one node faces a significant security bottleneck problem, while querying multiple blockchain nodes exacerbates the first problem. What's worse, some network attacks will isolate the device from legal blockchain nodes when the device needs to check a certificate. It is intuitive but infeasible to let every device maintain the entire ledger of blockchain, due to their limited resources, especially storage.

In order to realize a low-latency and reliable authentication, it is recommended to use lightweight blockchain nodes [9]. Rather than maintains the entire data, a lightweight node only stores the block header of each block, thus helps it to check transaction status locally with the method of *simplified payment verification* (SPV). The property of lightweight node has made blockchain a popular option for IoT, industrial environments, etc.. However, before utilizing it to manage identities, several important issues need to be taken into account. One critical problem is revocation. Any revoked certificate (denoted as $cert_A$) has two pieces of "valid" SPV auxiliary information on verifier's sight: one is related to a transaction for registering $cert_A$, and the other is related to the revocation. When the owner U_A wants to authenticate to a lightweight verifier, it can easily succeed in this forge, unless U_A consciously submits the SPV auxiliary information for revocation. Although some blockchain-based identity management schemes have proposed effective revocation method [10], [11], [12], [2], they do not work for lightweight node, as they require the verifier to be informed of all transactions in the ledger. Therefore, the biggest challenge in achieving secure and low-latency authentication is to establish a *reliable and efficient lightweight node-aware revocation mechanism*.

Aiming at this challenge, this paper proposes a lightweight node-support identity management scheme, whose authentication well resists the revoked users' forging. The proposed work uses a short Bloom filter to record the revoked certificates; and additionally leverages a pairing-based accumulator to achieve a precise certificate validity check, which is our proposed mechanism (called AcBF, Accumulator-assisted Bloom filter)

to fix the *false positive situation* of pure Bloom filter with slight storage and latency cost. Furthermore, according to the expected system scale, this paper studies the parameter selection of the bloom filter in our novel ACBF mechanism, in order to minimize the system cost for the revocation event.

Different from related work, in order to realize a trust revocation-informed block header update, even in an insecure channel (e.g., the downlink channel is controlled by a malicious party, or the node is attached to a dishonest full node to get headers), we construct a new certificate management scheme based on our AcBF mechanism. In the proposed scheme, with our designed revocation transaction and small-volume header structure, the lightweight verifier can be aware of the revocation events with only the block header, thus achieves trust local query to improve the efficiency and reliability.

In summary, this paper makes the following contributions:

1) We propose a lightweight on-chain certificate management, where the authentication phase incurs negligible overhead to validate the status of the certificate. Specially, the certificate revocation list (CRL) is composed of an accumulator-assisted Bloom filter, thus the filter length is drastically shortened without false detection.
2) We enable trust and low-latency authentication by facilitating the deployment of lightweight nodes, which makes them be aware of revocation event totally according to the block header. Compared to the previous researches, we further implemented the protocol and proposed the revocation consensus solution, allowing the lightweight node to query identity status locally and timely, in addition significantly decreasing the latency.
3) We implement AcBF in our blockchain system, and evaluate its performance in terms of computation and storage overhead. Results show that AcBF informs every revocation to lightweight nodes, and the overhead is extremely small on both certificate owners and verifiers, thus quite suit scenarios like IoT and IoV.

II. Preliminaries

A. Bilinear Pairing and q-SDH Assumption

Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be 3 groups of a same prime order p. A pairing map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ satisfies:

- *Bilinearity:* for all $(x, y, g_1, g_2) \in \mathbb{Z}_p^2 \times \mathbb{G}_1 \times \mathbb{G}_2$, $e(g_1^x, g_2^y) = e(g_1, g_2)^{xy}$;
- *Non-degeneracy:* if g_1, g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, $e(g_1, g_2)$ generates $\mathbb{G}_T$;
- *Efficiency:* It is efficient to compute $e(u, v)$ for all $(u, v) \in \mathbb{G}_1 \times \mathbb{G}_2$.

According to the relationship of $\mathbb{G}_1$ and $\mathbb{G}_2$, there are three types of pairings [13]. In Type 1, $\mathbb{G}_1 = \mathbb{G}_2$; In Type 2, $\mathbb{G}_1 \neq \mathbb{G}_2$, but there is a unidirectional homomorphism $\phi : \mathbb{G}_2 \to \mathbb{G}_1$; An in Type 3, no PPT homomorphism algorithm exists between the two groups. Overall speaking, Type 3 pairings are more secure and efficient to map, which is used in this paper.

Consider the q-Strong Diffie-Hellman Problem (q-SDH) [14] as follows: given a tuple $(g_1, g_2, g_2^{\gamma}, g_2^{(\gamma^2)}, \ldots, g_2^{(\gamma^q)})$ as input, where $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$, and $\gamma \in \mathbb{Z}_p$, outputs a pair $(g_1^{1/(\gamma+x)}, x)$ where $x \in \mathbb{Z}_p$. We say that an algorithm $\mathcal{A}$ has advantage ϵ in solving q-SDH if it has probability ϵ to return the valid pair.

Definition 1 (q-SDH Assumption): No PPT algorithm $\mathcal{A}$ has a non-negligible advantage ϵ for q-SDH problem.

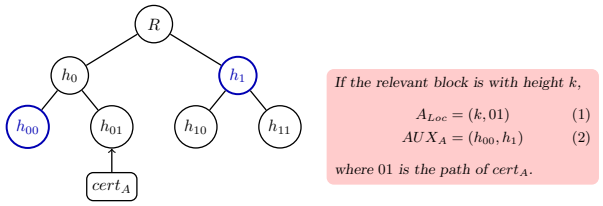


Fig. 1: Instance of merkle hash tree (MHT)

B. Bloom Filter

Bloom filter (or BF for short) is a space-efficient data structure to record a set, in order to support membership queries [15], [16]. BF is an array of m boolean bits, whose initial status is all zeros (denoted as 0^m). Another parameter k is the number of independent hashes, each of which maps a member to a random bit of BF. The follows describe the functions of BF, which will be used in the later context.

- $BF.Add(x)$: To add an element x to BF, it respectively hashes x with the k hashes of BF, and sets each of the mapped bit to 1.
- $BF.Check(x)$: To query if x has been added, it uses the k hashes to map x to a set of bits. If any of them is 0, it returns False; otherwise, it returns True.

Especially, the $BF.Check()$ gives a False result with no doubt; whereas the True result has the potential to happen when the tested element has not been added, which means a false positive situation.

C. Merkle Tree and Simplified Payment Verification (SPV)

Merkle hash tree [17] is essentially a binary tree where the leaf node is labeled a hash of a data block, and the non-leaf node is labeled with the hash of its child nodes. Such structure allows efficient verification of any large data. As shown in Fig. 1, $cert_A$ is labelled with the 2nd leaf of MHT, whose path is "01" (0 for left child and 1 for the right).

Simplified payment Verification (SPV) is a method to check if one data block is in a MHT, when the verifier only knows the root node. To realize the verification, SPV requires the prover side to offer necessary auxiliary information to the verifier, which means the values of **all sibling nodes from the hash of proved data block to the root**. Return to Fig. 1, the auxiliary information of $cert_A$ is as: $AUX_A = (h_{00}, h_1)$. Since each one obtaining tuple $(R, cert_A, AUX_A)$ can check if:

$$R \stackrel{?}{=} h(h(h_{00}, h(cert_A)), h_1) \quad (1)$$

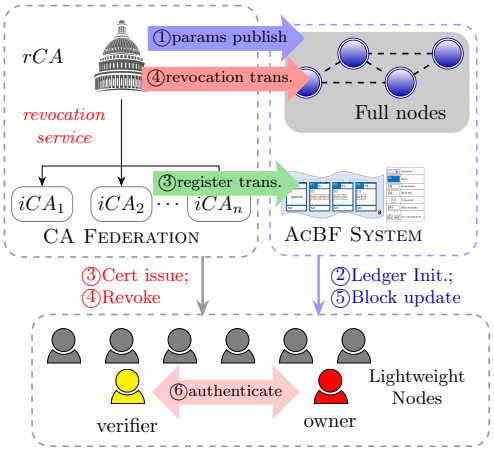


Fig. 2: Blockchain-based Identity System Architecture

III. System and Security Model

As shown in Fig. 2, the proposed scheme works in a communication system with the following 3 kinds of entities: multiple multiple certificate authorities (CAs), a set of blockchain full nodes and many lightweight nodes. The follows describe each of the entities.

A. Entity Description

1) Certificate Authorities: The CAs play the similar role as those in traditional PKI systems, which are responsible for issuing and revoking certificates for other entities. AcBF sepereates multiple CAs into two categories: one revocation CA (rCA) for certificate revocation, and multiple issuer CAs (iCA) for issuing affairs.

The rCA should be responsible for parameter determination (as ① in Fig. 2), and revoke on-chain certificates by ④submitting revocation transactions. Different from PKI, the iCAs are not fully trusted, they issue on-chain certificates to other entities by ③submitting registration transactions.

It should be mentioned that, it is easy to fully decentralize AcBF by realizing a threshold revocation among multiple rCAs. There are exiting cryptographic solutions such as [18] to be leveraged. For the clarity of introduction, this paper just uses a one-rCA model.

2) Full Nodes: They are blockchain nodes storing all blocks including transactions, and taking part in every-round consensus. Our scheme does not depend on a specific consensus method. Full nodes provide the following services: they gather transactions from CAs to generate blocks, ⑤ broadcast latest block header to lightweight nodes, and ② help lightweight node to initiate local ledger, when the lightweight nodes newly enters the system just then.

3) Lightweight Nodes: They are resource constrained devices from the perspective of storage and bandwidth. Thus they only store block header for transaction validation. It keeps up with the blockchain status by receiving newly generated blocks from at least one reachable full nodes.

The lightweight nodes will communicate with each other, thus should ⑥ authenticate and convince the opponent that its identity is valid. On this aspect, each lightweight node play two roles: it is a verifier based on its lightweight ledger; and also a certificate owner, that should maintain its authentication parameters, including secret key and auxiliary information.

B. Security Assumption

In our system, the rCA, as a core entity, is totally trusted. It securely maintains the security of secret keys and honestly generates the revocation transactions. The iCAs are not strictly honest; whereas, any entity can determine the certificate credibility according to the trust level of relevant iCA. Luckily, the blockchain is able to construct such trust well.

Any single full node may not be honest: it may try responding spoofing transaction or hiding revocation if it can benefit. However, the overall blockchain system is assumed safe and live based on the consensus mechanism; it is hard to propagates a non-consensus block header due to the assumption of selected consensus mechanism. For instance, in Proof of Work (PoW), malicious full nodes cannot succeed in puzzle solving game, thus the lightweight node will easily detect false version when it gets a longer fork.

The lightweight nodes cannot be trusted as certificate owners: they try to forge a valid certificate status, even if it has been revoked. In addition, malicious nodes will even forge a full node to broadcast false header to others and control the communication links. As a verifier, we assume that they will try their best to update their local ledger to the correct and latest version.

This paper makes further assumption about the communication links of lightweight node: 1) during long-lifetime interaction, the link is unstable and insecure; 2) in its initialization phase, it has adequate resource to update its local ledger to the latest and correct version, including other variables, e.g., Bloom filter and accumulator.

C. Design Goals

Based on the security assumptions, AcBF should achieve the following properties as an efficient and secure scheme:

- *Revocation-detectable authentication.* Only user's with valid and non-revoked certificate can authenticate successfully, faced with lightweight node.
- *Lightweight node-aware revocation.* No revocation event should be overlooked by a lightweight node, even in an untrusted environment.
- *Low-latency authentication.* Latency should be strictly restricted regardless of any situation.
- *Slight impact.* The above properties should be achieved without bringing much burden to honest entities, e.g, rCA and lightweight nodes.

IV. Construction of AcBF

A. Revocation-Aware Block Format

The block structure in this paper is basically similar to that of Bitcoin, except for the consideration of lightweight node awareness. Firstly, the storage cost of block header should be as short as possible with efficient query and sacrificing no

TABLE I: Block Header Format

Field	Size	Description
Height	4 bytes	An incremental integer to identify this block
Previous Hash	28 bytes	A hash of previous block
Flag	1 bit	Indication of revocation
MHT	28 bytes	Hash of all transactions in this block
Consensus	*variable*	Some method proving the validity of block

security feature; Secondly, the block header alone can prompt every certificate revocation clearly.

Taking into account the two issues, we redesign the 1) fields of block header, and 2) the tree structure of transactions. The header is a fixed length data, such that an unstructured storage can support fast addressing based on block height. The necessary fields are depicted in Table I. Especially, as our later implementation uses 224-bit hash function and curves, the presented sizes of Previous hash and MHT (hash of merkle hash tree) are 28 bytes. The Flag field is a main difference for the second consideration: it is a boolean variable to notify the holder whether revocation events occur in this block.

The consensus field depends on the used consensus method, e.g., it is a nonce for puzzle-like consensus method like PoW, or a digital signature for permission-based consensus.

All registered certificates (regarded as transactions) are structured in merkle hash tree as Section II-C, whose hash of root node is represented as R_c. There is also a slight difference for revocation awareness: if there is no revocation, R_c is just the value of MHT (denoted as R_a); otherwise, MHT is assigned the hash of R_c and the signature of revocation transaction in this block (denoted as R_x). For certificates registered in this block, their SPV auxiliary information includes R_x in such case. For the clarity of description, Table II lists the main notations and their meanings.

TABLE II: Notations

Notation	Meaning
Δ	Public value of the accumulator
BF	Bloom Filter
$\mathbb{R}_x$	Detailed revocation list containing every item
$\mathbb{R}$	A list of the gathered revocation in this round
rx	Content of a revocation transaction
R_x	Signature of rx by rCA
R_a	The value of MHT filed in the block
R_c	The root hash of the certificates
$cert_i$	Certificate of user U_i
i_h	The hight of $cert_i$'s registered block
i_p	The path of $cert_i$ in the MHT of the block
i_{Loc}	$i_{Loc} = (i_h, i_p)$ is the location of $cert_i$
AUX_i	The auxiliary information of $cert_i$ for MHT
W_i	Witness of $cert_i$ as an accumulator member
T_i	A flag indicates whether U_i should use accumulator

B. System Procedures

1) Initialization: The rCA initiates the system by generating a bilinear tuple $PP = \{p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e\}$ and two hash functions $H : \{0,1\}^* \to \mathbb{Z}_p$, $H_1 : \{0,1\}^* \to \mathbb{G}_1$. It selects its master secret key $\gamma \in \mathbb{Z}_p$, accumulator value $\Delta \in_R \mathbb{G}_1$, and parameters of Bloom filter (m, k) as Section II-B, based on the expected system scale as Section IV-C. It further generates an empty revocation list $\mathbb{R}_x$. The system parameters are published as

$$\left\{PP, \Delta, Y = g_2^{\gamma}, (m,k), H, H_1\right\}$$

Blockchain full nodes are gathered by generating a genesis block B_0, whose header's format follows Table I, with all-zeros fields, except the consensus.

The iCAs are registered in the blockchain as Section IV-B2. As a distributed system, iCA's certificates should be registered in self-sovereign manner, while our AcBF also supports issuing by rCA, with its key pair (γ, Y) under any signature algorithm. Note that, iCA's certificate status is queried like other entities', hence the later content will not specifically describe the iCA's status check in authentication phase.

Every lightweight node initiates its ledger by updating the block headers from selected full nodes, like other lightweight node-supported schemes. In addition, it updates the Bloom filter BF and accumulator value Δ. As the full nodes are not trusted, the BF and Δ are checked from the blocks with revocation transactions as depicted in Section IV-B3.

2) Certificate Register: From an arbitrary trusted iCA, each entity U_i applies this phase to register an on-chain certificate $cert_i$. The certificate in AcBF is basically similar to that in X.509, except that AcBF does not need the globally unique sequence number. Instead, AcBF uses its unique location $i_{Loc} = (i_h, i_p)$ in blockchain to identify the certificate, where i_h is denoted as the height of registered block, and i_p is its path in MHT. Fig. 1 gives an instance of a certificate's location.

U_i securely stores a secret key sk_i, whose relevant public key pk_i is included in $cert_i$. The U_i should wait for this certificate to be gathered with other transactions (including registration and revocation transactions). After the consensus and broadcast of the block as Section IV-B5, this phase is completed with U_i getting its i_{Loc} and the auxiliary information of MHT AUX_i. It stores the following parameters locally:

$$(cert_i, i_{Loc}, sk_i, AUX_i, T_i = 0),$$

in which, $T_i = 0$ means that U_i initially can authenticate directly, without the usage of accumulator.

3) Revocation: This phase consists of I) execution by rCA, and II) reaction by other entities, especially the lightweight nodes.

I) To execute revocation, rCA prepares a temporary revocation list $\mathbb{R}$ to gather revocation request from legal iCAs and certificate owners. After validation of each request, rCA appends the location i_{Loc} of the certificate to $\mathbb{R}$ until it invokes Algorithm 1 to generate revocation transaction.

Especially, AcBF proposes an efficient accumulator method based on [19], including the following three functions:

- *Add* makes the input value x_i as the member of accumulator, by outputting W_i as its witness;
- *Del* removes the member x_i by updating the value Δ;
- *Update* helps the other members x_i to update their witnesses with the input parameters.

Algorithm 1 Revocation Procedure of rCA

Require: $\mathbb{R}$, $\mathbb{R}_x$, BF and Δ
Output: Revocation Transaction (rx)
1: $L \leftarrow \emptyset$
2: $BF_{tmp} \leftarrow 0^m$ ▷ Empty Bloom filter
3: **for all** $x \in \mathbb{R}$ **do**
4: **if** $BF.Check(x)$ **then**
5: $\Delta \leftarrow Del(\Delta, x)$ ▷ as Algorithm 2
6: append (x, Δ) to L
7: **else**
8: $BF_{tmp}.Add(x)$
9: **end if**
10: **end for**
11: $BF \leftarrow BF \vee BF_{tmp}$, $\mathbb{R}_x \leftarrow \mathbb{R}_x \bigcup \mathbb{R}$
return $rx \leftarrow (L, \mathbb{R})$

Algorithm 2 Accumulator

Require: $\gamma, \Delta, Y = g_2^{\gamma}$
1: **function** ADD(Δ, x_i)
return $W_i \leftarrow \Delta^{1/(x_i+\gamma)}$
2: **end function**
3: **function** DEL(Δ, x_j)
return $\Delta \leftarrow \Delta^{1/(x_j+\gamma)}$
4: **end function**
5: **function** UPDATE($(x_i, W_i), \Delta, x_j$) ▷ x_j is revoked
return $W_i \leftarrow (W_i/\Delta)^{1/(x_j-x_i)}$
6: **end function**

Note that, as x_i in AcBF is a location format, the real input uses $H(x_i)$ to replace x_i.

The detailed execution of these functions in this paper are illustrated in Algorithm 2. Especially, the Update is correct as

$$\begin{aligned} W_i =& (W_{i(old)}/\Delta)^{1/(x_j-x_i)} \\ =& (\Delta_{old}^{\frac{1}{x_i+\gamma}-\frac{1}{x_j+\gamma}})^{1/(x_j-x_i)} \\ =& (\Delta_{old}^{\frac{1}{x_j+\gamma}})^{\frac{1}{x_i+\gamma}} = \Delta^{1/(x_i+\gamma)} \end{aligned} \tag{2}$$

where $W_{i(old)}$ and Δ_{old} are the U_i's valid witness and accumulator value before x_j's Delete function, respectively.

Then, rCA submits rx along with the signature R_x to the blockchain nodes, using its secret key γ. AcBF does not restrict the signature algorithm for revocation transactions, in fact any short and secure method can be implemented, such as BLS [20] signature. In BLS, the format of signature R_x and its verification are as (3) and (4), respectively.

$$\sigma = H_1(rx)^{\gamma} \tag{3}$$

$$e(\sigma, g_2) = e(H_1(rx), Y) \tag{4}$$

II) When a lightweight node U_i invokes the revocation reaction by the notification of Flag as Section IV-B5, it updates the maintained BF and Δ with the revocation transaction rx. Parse rx as $(L, \mathbb{R})$ and executes Algorithm 3.

Algorithm 3 Revocation Reaction by Lightweight Node U_i

Require: $BF, i_{Loc}, L, \mathbb{R}, W_i$ if $T_i = 1$
Output: Δ
1: **if** $L \neq \emptyset$ **then**
2: Update Δ as the last tuple of L
3: **end if**
4: **for all** $x \in \mathbb{R}$ **do**
5: $BF.Add(x)$
6: **end for**
7: **if** $T_i = 1$ **then**
8: **for all** $(x_j, \Delta_j) \in L$ **do**
9: $W_i \leftarrow Update((i_{Loc}, W_i), \Delta_j)$ ▷ Line 5 of Alg. 2
10: **end for**
11: **else if** $BF.Check(i_{Loc})$ **then**
12: Invoke $W_i \leftarrow Add(\Delta, i_{Loc})$ with rCA
return W_i and $T_i \leftarrow 1$
13: **end if**

Worth noting that, Line 12 of this algorithm focuses on the case a legal user will be innocently checked as revoked one if only with Bloom filter. Whereas, AcBF uses the accumulator Δ to fix this problem. The rCA checks the follows: Firstly, U_i's unique location is not in current $\mathbb{R}_x$, such that it is a valid user; Secondly, checks

$$BF.Check(i_{Loc}) = 1,$$

which indicates that it is innocent in BF. If the above hold, rCA responds the request; otherwise, it aborts.

4) Authentication: When U_i needs to authenticate itself to a lightweight verifier V, it should sign a certain message or a challenge with its sk_i. This paper focuses on the issue that how U_i convinces V that the relevant pk_i is indeed associated with a valid on-chain certificate.

The certificate validation message includes the certificate $cert_i$, its registered location i_{Loc} (including i_h and i_p), the auxiliary information of MHT AUX_i. If $T_i = 1$, then the witness W_i is also in the message. The entire message is as

$$\sigma = \Big(Sig(m; sk), cert_i, i_{Loc}, AUX_i; W_i \text{ if } T_i = 1 \Big)$$

Upon the recept of σ, V checks the follows:

- *Existence*: checks the tuple $(R_a, cert_i, AUX_i)$ as (1), where R_a is the MHT field of the i_h-th block.
- *Non-revocation*: If σ does not contain W_i, checks

$$BF.Check(i_{Loc}) \stackrel{?}{=} 0;$$

Else, checks

$$e(W_i, g_2^{H(i_{Loc})} \cdot Y) \stackrel{?}{=} e(\Delta, g_2) \tag{5}$$

- The signature is validated with public key in $cert_i$.

If the above is all passed, V is convinced that U_i is valid. Otherwise, the authentication is failure.

5) Block Generate and Broadcast: The blockchain full nodes collect certificate register transactions from muliptle iCAs, and at most one revocation one rx (with signgature R_x). According to the selected consensus mechanism, the nodes organize the transactions to MHT, according to Section IV-A, as well as the block header encapsulation: the Flag field is set to "1" or "0", depending on whether this block contains an rx. Let the height of this newly generated block be h_n.

After this block is generated, other full nodes check its validity including the follows:

- *Transactions:* All transactions are valid, especially, R_x is the correct signature of rx with Y as Eq. (4).
- *Structure:* $h_n = h_c + 1$, where h_c is the largest height of stored block; rx exists if Flag is "1"; and all hashes are valid, including the value of Previous hash field.

After consensus, the block header, as well as rx, R_x and R_c (the MHT root of only certificate registration transaction) are transmitted to every lightweight node. Upon the recept of the header, each lightweight node checks as:

1) If Flag is "1" without rx, the lightweight node should request rx as well as R_c. Otherwise, finishes the check.
2) Check R_x is the signature of rx and $H(R_c, R_x) \stackrel{?}{=} R_a$, which is the value of MHT field of this block.

Whether full or lightweight nodes, if the Flag is "1", then invokes Algorithm 3 with the verified rx, to update local BF, accumulator value Δ and optionally its witness. After that, the lightweight node removes rx, R_x and R_c.

C. Parameter Determination for Bloom Filter

The length of Bloom filter in AcBF can be dramatically decreased compared with other schemes for revocation. Whereas, a shorter filter length causes larger member size of the cryptographic accumulator, and ultimately results in potentially larger system overhead for the revocation. To optimize the efficiency, this section gives a quantitative study of parameter selection.

Let n, δ be the possible maximal number of registered and revoked users, respectively. The remaining factor is system effect of user revocation: when revocation is executed in one consensus round, the expected affected users should be limited to θ, which can be measured as the member amount of Accumulator Δ. The measurement is feasible as:

- Larger member amount in Δ causes larger probability that a user to be revoked has already in Δ and more members to update their witness.
- The probability that innocent users apply for accumulator member affects the increase rate of θ. Thus, θ also implies the effect when revocation is executed just in Bloom filter.

Assume m is the calculated hash length with k hash functions in the Bloom filter. With δ users already being revoked, the probability can be measured as follows, that an legal user is erroneously claimed as a revoked one:

$$\Pr(1) \simeq (1 - e^{-k\delta/m})^k \tag{6}$$

The parameter k can be individually optimized to minimize $\Pr(1)$ as $k = \frac{m}{\delta} \ln 2$. Then, for n valid users, the expected affected amount (limited to θ) is

$$\theta = n \cdot (1 - e^{-k\delta/m})^k = 0.5^{\frac{m}{\delta} \ln 2} \cdot n \tag{7}$$

From (7), the parameters of Bloom filter are determined as

$$m = \frac{\ln \frac{n}{\theta} \cdot \delta}{(\ln 2)^2} \simeq 2.08\delta \cdot \ln \frac{n}{\theta} \tag{8}$$

$$k = \lfloor \ln \frac{n}{\theta} / \ln 2 \rceil \simeq \lfloor 1.44 \ln \frac{n}{\theta} \rceil \tag{9}$$

V. Security Analysis of AcBF

A. Sound Authentication and Reliable Revocation

Theorem 1: The AcBF identity management achieves sound and revocation-detectable user authentication for the lightweight-node verifier.

Proof: A user without a registered certificate in public ledger (even without a revoked identity) will not forge an authentication due to the traditional certificate technique with trust SPV method. Thus this proof mainly focuses on revoked user. Formally, this theorem holds if a revoked user $\mathcal{A}$ wins the following challenges with negligible advantage.

1) With the knowledge of accumulator value Δ, as well as any other user's accumulator pair from authentication message (i_{Loc}, W_i), $\mathcal{A}$ tries to forge its witness.
2) When $\mathcal{A}$ goes through a revocation based on member delete in Δ, it tries to forge a new witness value for the updated Δ.
3) When $\mathcal{A}$ is revoked in block j, it tries to hide the revocation fact during the broadcast of block header.

The first 2 challenges require a secure cryptographic accumulator, and a strong blockchain protocol is in need for the last challenge. Hence, the demonstration of Theorem 1 can be derived from the following 2 lemmas. ■

Lemma 1.1: If q-SDH assumption holds, the revocation method based on accumulator mechanism in Algorithm 2 is sound against revoked identities.

Proof: The accumulator algorithm was basically proved secure under q-SDH assumption in [19], whereas, AcBF makes two modifications. Firstly, we remove the zero-knowledge proof (ZKP) to accelerate the authentication phase as (5); Secondly, *Add* function inserts an arbitrary member value without changing the accumulator public value Δ.

In fact, non-ZKP authentication only helps the adversary $\mathcal{A}$ to get the member secret (or the unique location i_{Loc} in AcBF) and its witness W_i, compared with standard algorithm. Whereas, the i_{Loc} is no longer secret in AcBF, and $\mathcal{A}$ cannot learn the information of the secret key sk_i, which is relevant to the public key in the transaction recorded in i_{Loc}. Thus, as long as the consensus mechanism is safe (to ensure that every i_{Loc} is unique between certificates), the additional information in (5) brings no advantage to adversaries. On the other hand, we analyze the advantage from cryptographic perspective. A tuple $(\Delta, x, W = \Delta^{1/(x+\gamma)})$ can be reduced to $(B = h^{x+\gamma}, x, h)$, which leaks no knowledge on γ. The final mechanism is that we

use naturally unique location to replace the sequence number in certificate. As a result, even the iCA is not totally trust, which might issue certificates illegally, it cannot issue a certificate with the same key index with another exiting one, since the location's uniqueness is inherent.

The second modification makes our accumulator be a stand q-SDH tuple: Given as system with a secret key γ, and public group elements $\Delta \in \mathbb{G}_1$, $(g_2, Y = g_2^{\gamma}) \in \mathbb{G}_2$. The hash of member value and its witness $(H(i_{Loc}), W_i)$ is identical to the tuple $(x, \Delta^{1/(x+\gamma)})$.

The Member *Del* and *Update* function in Algorithm 2 is also identical to key revocation in BBS signature [14]. An intuitive description about the revocation capability is that the revoked user U_j should deal with the problem of zero denominator to update its witness, as $x_i = x_j$. Formal reduction from BBS signature to q-SDH assumption is given in [14]. As our modified accumulator can be reduced to BBS, it is semantically secure under q-SDH assumption. ∎

Lemma 1.2: If the consensus protocol of the used blockchain is safe, the lightweight node is aware of every revocation when its ledger has been updated to the correct status.

Proof: When a certificate is successfully revoked in a block (e.g., of height i), the Flag field of its header should be *True*. As long as the consensus protocol has a strong safety property, the lightweight node V will not accept the i-th block header with Flag field *False*.

With the above result, any revocation transaction should be sent to the V with the signature R_x and auxiliary information R_c, where R_c is the root value of all certificate registration transactions in this block. No adversary can hide or modify even one revocation item in this scenario due to the unforgeable signature and hash strength, even by forging a full node or controlling the downlink channel. ∎

B. Accurate Revocation Policy

Note that every identity's T_i is a logical and locally stored boolean parameter. In the whole procedure of AcBF, it is not transferred in any cases. Analysis of revocation soundness in Theorem 1 has already shown that authentication phase will detect every certificates precisely without explicit T_i indication. This section discusses if the rCA can accurately deal with revocation algorithms. Here, accuracy means two sides, completeness and low-impact. The first requires that every revocation should be added in AcBF revocation data structure; the second tries to bring in as least accumulator members as possible.

On the one side, as long as rCA executes Algorithm 1 correctely, the revocation will executed completely. Since every newly revoked certificate in $\mathbb{R}$ is added in the short Bloom filter. We further check if every element in $\mathbb{R}$ that their BF check has already been *True* with the old-version BF, which indicates that the certificate may already be an accumulator member (e.g, it was once an innocent user, and applied the member). The member Add function requires an efficient check, such that the relevant certificate's further revocation will succeed by just adding it to the Bloom filter. As a summary, we design a lightweight algorithm for rCA to achieve a complete revocation.

On the other side, we use a temporary Bloom filter BF_{tmp} in Algorithm 1 to minimize the impact: the accumulator Del function is not invoked in the case, when the certificate is checked false in current BF of other entities, but is checked revoked in the rCA. The issue is intuitive, but without the proposed revocation policy, it will be difficult or inefficient to realize this goal (e.g. additionally maintain a list of all accumulator members).

C. Lightweight Overhead

AcBF achieves this property on two aspects. Firstly, a lightweight node requires no interaction with full nodes to validate any certificate's status, regardless of any cases, thus achieves extremely low-latency authentication. Secondly, it brings in slight impact on the system to maintain the reliable revocation. This section mainly focuses from the security perspective that AcBF achieves the trust authentication with slight cost. Detailed overhead result is shown in Section VI.

On the first aspect, we use a short Bloom filter to record the revocation list, while use accumulator to fix its false positive situation: a wronged user applies a member of accumulator with effective check, thus achives fast authentication with an efficient member check as (5). Without this method, when a user is checked as revoked, it requires the verifier to request relevant transaction from trusted full nodes to judge the real status. Furthermore, only wronged certificate needs additional check based on accumulator, the average latency is still similar to no-revocation method.

On the second aspect, all entities cost negligible burden besides the basic block header update. 1) As a lightweight verifier, it only needs a short BF structure and an accumulator value to enable its validation ability. All revocation transactions should be erased after the time when verifier has update its BF and Δ. 2) As an identity owner, it basically maintains the location and auxiliary information of its registered certificate. 3) Even as accumulator member, the *Update* function only occurs when a user in accumulator is revoked, the probability is quite small. From the sight of a uniformly selected identity, its impact probability $\Pr(E)$ faced with a random certificate revocation can be expressed as

$$\Pr(E) = (\theta/n)^2, \tag{10}$$

where θ is the member size of current accumulator, and n is the amount of registered user.

In addition, compared with revocation mechanisms based on smart contract or MHT, every lightweight node can update its witness with the public revocation transactions locally, thus costs little even in any low-probability matters.

VI. Performance Evaluation

We implement AcBF and other compared schemes in C++ program with PBC library 0.5.14. The pairing and elliptic curves are MNT224 and SECP224r1, respectively. Bloom filter is realized with bitarray structure in Python 3.11.3. The CAs

TABLE III: Performance Comparison with Related Schemes

Scheme	Lightweight node *supp.*	Revocable	Revocatin mechanism	Additional method	Revocation Impact on Verifier	Owner	System	Issues faced with lightweight node
CertChain [2]	✗	✓	Bloom Filter	*n/a*	heavy	no	medium	latency even with large BF
CertLedger [21]	✗	✓	MHT	*n/a*	heavy	heavy	heavy	any affair affects system
ScalaCert [10]	✗	✓	Chameleon Hash	active check	medium	medium	medium	revoked *trans.* has "valid" SPV
PROCESS [22]	✗	✓		CGBF	medium	medium	medium	*ditto*
Jia *et al.* [12]	✗	✓		Accumulator	medium	medium	medium	*ditto*
Yu *et al.*[23]	✗	✓	Accumulator	*n/a*	light	heavy	medium	any revocation triggers update
SmartDID [9]	✓	✗	*n/a*	*n/a*		*n/a*		*n/a*
Proposed	✓	✓	AcBF	Flag	light	light	light	*short BF* & $1 \times \mathbb{G}_1$

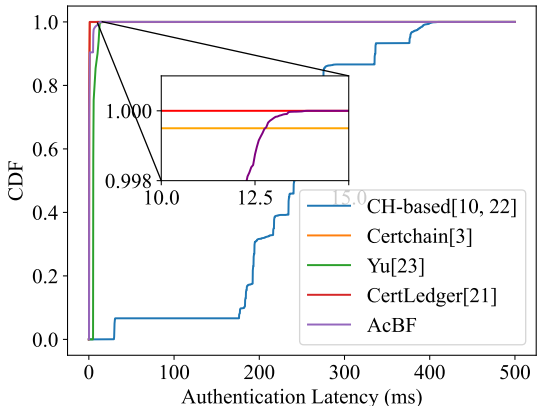


Fig. 3: CDF of Authentication Latency

are built on Ubuntu 20.04 platform by PCs with 2.5GHz Intel Core i7-11700, and other entities are built in Raspberry Pi platforms. The blockchain application is realized with Flask framework to realize network interactions, which is based on a public repository in [24]. Lightweight nodes communicate with each other by a one-hop WiFi channel, while full nodes connect distinct AS from the lightweight ones. The registered certificates in each block are as Poisson distribution with parameter $\lambda = 5,000$, while the revocation is the same distribution with $\lambda = 500$. We execute the system for 100 rounds to get the experiment results.

We briefly compare some representative work as shown in Table III in terms of critical factors for identity management. The detailed analysis are proposed in our later content. Worth noting that, we tried our best to enhance the compared schemes to support lightweight nodes in our later evaluations.

A. Authentication Latency

The latency measures the duration from the time a verifier gets the authentication message to the time it completes the verification. For all schemes in comparison, we use ECDSA to model $Sig(m; sk)$. The cumulative distributions of verification time for all 500,000 certificates are depicted in Fig. 3.

Certledger [21] achieves the shortest authentication latency: it manages all legitimate certificates in MHT, thus only needs $\log n$ hashes to check each certificate. All Chameleon hash (CH-based) schemes [10], [22] need the largest latency, since all revoked certificates still have "valid" SPV information, it requires the lightweight node to rely on reliable full nodes to judge the certificate status.

The accumulator method helps [23] achieve low latency. It should be mentioned that in [23], the latency should be larger since it requires zero-knowledge proofs for member validation. By using Bloom filter, CertChain [2] achieves high efficiency for most of time, but requires unbearable round trip time with full nodes in the false positive situation.

The proposed AcBF needs less than 1 ms in most of the cases (90%); in other cases, the accumulator member check in (5) requires about 4 ms on average, even the worst case will be finished in 12 ms. Considering the average and worst cases, AcBF shows its significant advantage. Especially, the worst-case latency well suits many low-latency environments, e.g., IoV.

B. Storage on Lightweight Verifiers

We compare the storage cost on verifier's side, where the ledger volume is not included in the measurement. Since the CH-based systems [10] do not require data structure for revocation lists, they achieve the best performance in this term, although the CH-based mechanism cannot support lightweight node. CertLedger [21] and Yu *et al.* [23] only needs a MHT root hash and accumulator public value, respectively. Thus, we only compare AcBF with Bloom filter-based schemes [25], [22]. Fig 4 shows the storage cost versus expected revocation numbers.

In AcBF, the overhead includes m-bit bloom filter and a group element for accumulator value Δ; while the benchmarks only need a Bloom filter, which should be much large than that in AcBF. Strictly, the storage cannot be compared, as in Certleger, false positive situation should be in a negligible probability, such that BF is suggest to be over 400 kB in [25]. We just make the Bloom filter's tolerated error probability be 0.01% and 0.05%. For AcBF, the maximal affected user is set as $\theta/n = 0.1$ and 0.2.

Fig. 4 shows that AcBF only needs very little storage to maintain the short BF and accumulator value (less than 20 kB). It is a very lightweight cost compared with direct Bloom filter. It should be mentioned that Certchain uses DCBF (double Bloom filter, with every entry being an integer) for user management, which costs much larger cost. What is more, as the related work cannot bear the false positive situation, the parameter in this comparison is not adequate. Differently, AcBF designs

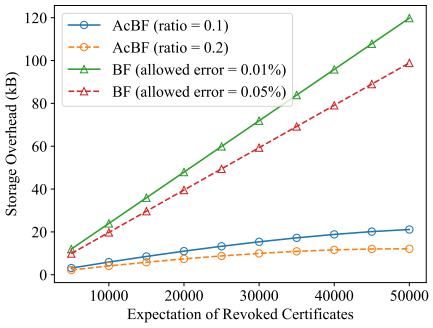
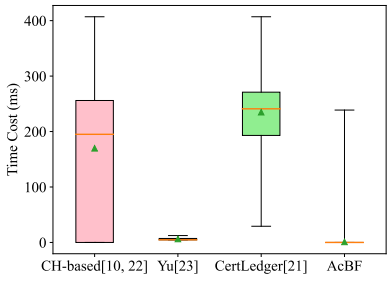
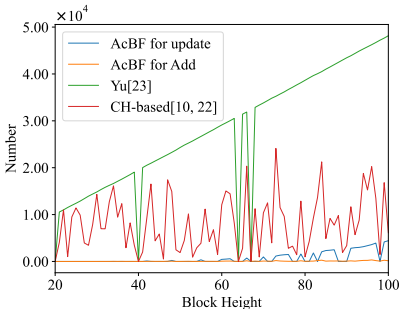


Fig. 4: Storage on Verifier for Revocation



(a) Time Cost for 1-Round



(b) Affected Number During Lifetime

Fig. 5: Evaluation Result on Owners' Reaction for Revocation

an efficient mechanism for this issue, thus the parameter is feasible, according to our former analysis.

C. Burden on Certificate Owner

It seems that AcBF shows the most significant advantage. Since the entity does not execute any task to maintain the legality of its certificate extremely high probability. Fig. 5 displays our measurement.

As shown in Fig 5a, schemes in [10], [21], [22] require full nodes to update their auxiliary information, thus require more time than other schemes. Especially, CH-based schemes require users in the certain blocks to undertake this burden, while in CertLedger [21], the MHT tree organizes all users, thus has the worst performance. Yu *et al.* [23]'s scheme achieves very short overhead since the update mechanism for accumulator only needs globally same parameters in the revocation transaction.

AcBF has even slighter burden since the accumulator member is only 10% of the user, which means accumulator update function occurs only with probability of 10%. AcBF requires interaction with rCA when it applies an accumulator member, but the probability is more rare. Fig 5b corroborates our analysis. As the work in [23] affects all certificate owners, while in [10], [22], it only affects certificates registered in some blocks. AcBF needs only few users to update its accumulator witness, and the impact for accumulator add is negligible.

VII. Related Works

Blockchain is a promising technology to devise reliable and decentralized certificate management. Fromknecht *et al.* [26] proposed CertCoin to ensure identity retention with cryptocurrencies. Compared with the traditional technique like CA-based PKI [4] and PGP method [27], the tamper free feature brought from blockchain tackles the gap between decentralization and certificate management. A totally decentralized model called SSI (self-sovereign identity) [28] was designed by the Sovrin Foundation in 2018. In SSI, users' identities are registered in blockchain by themselves. This idea has been realized in industrial implementations such as hyperledger indy [29] and uPort [30] attracted academic researches like SmartDID [9]. However, the full decentralized method does not suit practical systems, since it is complex to deal with trust and revocation.

Another way that blockchain improves identity system is to let the blockchain manage distributed CAs and certificate revocation list (CRL). Wang *et al.* [31] realized a transparent revocation by enabling OCSP (Online Certificate Status Protocol) on blockchain. Chen *et al.* [2] proposed CertChain, in which a dependability-rank based consensus organized muliptle CAs and uses double counting bloom filter (DCBF) to shorten the volume of CRL. However, Luo *et al.* [10] pointed out that DCBF still occupied too much storage resource on the chain. Chameleon hash [32] enables redactable blockchain such that revocation can be realized without CRL, thus many schemes were proposed, such as [10], [12], [33], [22]. Note that block redaction is difficult to make everyone aware, additional mechanism shold be proposed, e.g., random freshness check [10], RSA-based accumulator [12] and CGBF [22]. Much worse problem is that redactable blockchain does not suit lightweight node, as revoked certificate can still be "valid" since the block header has not varied.

Due to the similar distributed architecture, blockchain-based identity is promising to secure the future ad-hoc networks, such as IoT and IoV [6], [8], [9], [23]. Especially, Yin *et al.* [9] proposed SmartDID to support lightweight node deployment in identity system. Whereas, we found that the above revocation mechanisms could not work faced with a verifier only storing block headers. It is difficult for a lightweight node to cache and update CRL in time, especially when the downlink channel is not trust. Due to similar reasons, mechanisms in [10], [12] also failed. Some schemes leveraged MHT [21] or accumulator [23] to maintain all legitimate certificates, which seems to suit lightweight node since the verifier only needs to keep up with the root node or accumulator value. However, they will bring in unbearable burden on certificate owners' side (which are also thin devices), since it requires all owners to update their auxiliary information for any member registration and revocation.

VIII. Conclusion

In this paper, we have proposed AcBF to blockchain-based identity system for lightweight nodes, which is the first that realize certificate revocation with storage constrained ledger as a verifier. By systematically investigating the problems

of trust authentication scenarios for lightweight nodes, we constructed a security model for the identity system. With the usage of accumulator-assist short Bloom filter and relevant parameter selection mechanism, AcBF achieved the minimal volume to maintain the revocation data, as well as local verification regardless of any situations, thus offered a low-latency authentication. Furthermore, a new ledger structure and protocol has been proposed for lightweight node, which helps the node to be aware of every revocation event. As long as the consensus mechanism is safe, the above feature holds even in a insecure or unstable communication environment. With this feature, it checks every certificate's status totally according to its local ledger, thus achieves an extremely low authentication latency. The security proof and the implementation results have indicated that AcBF could achieved our expected properties.

ACKNOWLEDGEMENT

This work was supported by National Key Research and Development Program of China (2022YFB2702300), Research and Development Program of Department of Industry and Information Technology in Xinjiang Autonomous District (No. SA0304173) and Natural Science Foundation of China (Grant Nos. 62202290, 62302343).

REFERENCES

[1] Z. Zhao and Y. Liu, "A blockchain based identity management system considering reputation," in *Proceedings of the International Conference on Information Systems and Computer Aided Education (ICISCAE)*, 2019, pp. 32–36.

[2] J. Chen, S. Yao, Q. Yuan, K. He, S. Ji, and R. Du, "Certchain: Public and efficient certificate audit based on blockchain for tls connections," in *Proceedings of IEEE International Conference on Computer Communications*, 2018, pp. 2060–2068.

[3] Z. Yang, K. Yang, L. Lei, K. Zheng, and V. C. Leung, "Blockchain-based decentralized trust management in vehicular networks," *IEEE internet of things journal*, vol. 6, no. 2, pp. 1495–1505, 2018.

[4] D. Cooper, "Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile," *RFC5280*, 2008.

[5] C. Zhang, L. Zhu, and C. Xu, "BPAF: Blockchain-Enabled Reliable and Privacy-Preserving Authentication for Fog-Based IoT Devices," *IEEE Consumer Electronics Magazine*, vol. 11, no. 2, pp. 88–96, 2022.

[6] J. Cui, N. Liu, Q. Zhang, D. He, C. Gu, and H. Zhong, "Efficient and anonymous cross-domain authentication for iiot based on blockchain," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 2, pp. 899–910, 2022.

[7] A. Lei, H. Cruickshank, Y. Cao, P. Asuquo, C. P. A. Ogah, and Z. Sun, "Blockchain-based dynamic key management for heterogeneous intelligent transportation systems," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1832–1843, 2017.

[8] M. Singh and S. Kim, "Branch based blockchain technology in intelligent vehicle," *Computer Networks*, vol. 145, pp. 219–231, 2018.

[9] J. Yin, Y. Xiao, Q. Pei, Y. Ju, L. Liu, M. Xiao, and C. Wu, "Smartdid: a novel privacy-preserving identity based on blockchain for iot," *IEEE Internet of Things Journal*, vol. 10, no. 8, pp. 6718–6732, 2022.

[10] X. Luo, Z. Xu, K. Xue, Q. Jiang, R. Li, and D. Wei, "ScalaCert: Scalability-oriented pki with redactable consortium blockchain enabled "on-cert" certificate revocation," in *Proceedings of IEEE International Conference on Distributed Computing Systems*, 2022, pp. 1236–1246.

[11] Z. Wang, J. Lin, Q. Cai, Q. Wang, D. Zha, and J. Jing, "Blockchain-based certificate transparency and revocation transparency," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 1, pp. 681–697, 2020.

[12] M. Jia, J. Chen, K. He, R. Du, L. Zheng, M. Lai, D. Wang, and F. Liu, "Redactable blockchain from decentralized chameleon hash functions," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 2771–2783, 2022.

[13] S. D. Galbraith, K. G. Paterson, and N. P. Smart, "Pairings for cryptographers," *Discrete Applied Mathematics*, vol. 156, no. 16, pp. 3113–3121, 2008, applications of Algebra to Cryptography.

[14] D. Boneh, X. Boyen, and H. Shacham, "Short Group Signatures," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3152, pp. 41–55, 2004.

[15] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, jul 1970.

[16] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," *Internet mathematics*, vol. 1, no. 4, pp. 485–509, 2004.

[17] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Conference on the theory and application of cryptographic techniques*. Springer, 1987, pp. 369–378.

[18] R. Gennaro, S. Goldfeder, and B. Ithurburn, "Fully distributed group signatures," 2019.

[19] L. Nguyen, "Accumulators from bilinear pairings and applications," in *Proceedings of The Cryptographers' Track at the RSA Conference*. Springer, 2005, pp. 275–292.

[20] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," *Journal of cryptology*, vol. 17, pp. 297–319, 2004.

[21] M. Y. Kubilay, M. S. Kiraz, and H. A. Mantar, "Certledger: A new pki model with certificate transparency based on blockchain," *Computers & Security*, vol. 85, pp. 333–352, 2019.

[22] M. Jia, K. He, J. Chen, R. Du, W. Chen, Z. Tian, and S. Ji, "Process: Privacy-preserving on-chain certificate status service," in *Proceedings of IEEE International Conference on Computer Communications*. IEEE, 2021, pp. 1–10.

[23] Y. Yu, Y. Zhao, Y. Li, X. Du, L. Wang, and M. Guizani, "Blockchain-based anonymous authentication with selective revocation for smart industrial applications," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 5, pp. 3290–3300, 2019.

[24] OpensourceBooks, "blockchain," https://github.com/OpensourceBooks/blockchain, 2018.

[25] J. Chen, S. Yao, Q. Yuan, K. He, S. Ji, and R. Du, "Certchain: Public and efficient certificate audit based on blockchain for tls connections," in *Proceedings of IEEE International Conference on Computer Communications*, 2018, pp. 2060–2068.

[26] C. Fromknecht, D. Velicanu, and S. Yakoubov, "A decentralized public key infrastructure with identity retention," *Cryptology ePrint Archive*, 2014.

[27] J. Callas, L. Donnerhacke, H. Finney, and R. Thayer, "Rfc 2440: openpgp message format," *Internet Draft Rfc2440bis*, 1998.

[28] S. Foundation, "Sovrin™: A Protocol and Token for Self- Sovereign Identity and Decentralized Trust," *Sovrin*, no. January, pp. 1–41, 2018.

[29] H. Indy. (2023) Hyperledger indy 1.0 documentation. [Online]. Available: https://indy.readthedocs.io/en/latest/

[30] N. Naik and P. Jenkins, "uPort open-source identity management system: An assessment of self-sovereign identity and user-centric data platform built on blockchain," in *Proceedings of the IEEE International Symposium on Systems Engineering*. IEEE, 2020.

[31] Z. Wang, J. Lin, Q. Cai, Q. Wang, D. Zha, and J. Jing, "Blockchain-based certificate transparency and revocation transparency," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 1, pp. 681–697, 2022.

[32] G. Ateniese, B. Magri, D. Venturi, and E. Andrade, "Redactable blockchain–or–rewriting history in bitcoin and friends," in *Proceedings of European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2017, pp. 111–126.

[33] S. Xu, J. Ning, J. Ma, X. Huang, and R. H. Deng, "K-time modifiable and epoch-based redactable blockchain," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 4507–4520, 2021.