Hindawi
Security and Communication Networks
Volume 2022, Article ID 2069195, 15 pages
https://doi.org/10.1155/2022/2069195

*Research Article*

# CR-BA: Public Key Infrastructure Certificate Revocation Scheme Based on Blockchain and Accumulator

**Jingxue Xie**,[1] **Xinghong Tan**,[1] **and Liang Tan**[1,2]

[1]*School of Computer Science, Sichuan Normal University, Chengdu 610000, Sichuan, China*
[2]*Institute of Computer Science, Chinese Academy of Sciences, Beijing 100000, China*

Correspondence should be addressed to Liang Tan; jkxy_tl@sicnu.edu.cn

Received 29 April 2022; Accepted 11 June 2022; Published 31 July 2022

Academic Editor: Jiewu Leng



With the development of blockchain, many studies apply blockchain to certificate revocation. However, existing blockchain-based certificate revocation schemes have two shortcomings. First, the storage overhead on the blockchain is relatively large. Second, as the number of revoked certificates increases, the misjudgment rate of certificate status will increase accordingly, so a public key infrastructure implementation certificate revocation scheme based on blockchain and accumulators, called CR-BA, is proposed. First, CR-BA expands the certificate structure, adding a revocation factor and a smart contract account for accessing the blockchain in the certificate extension, which is filled by the CA when the certificate is generated. Then, when the certificate is to be revoked, CA generates the revocation fingerprint through the revocation factor and publishes it to the blockchain. Finally, when the user needs to verify the status of the certificate, CA calculates the revocation fingerprint according to the revocation factor on the certificate, then compares it with the existing revocation fingerprint on the blockchain, and returns the comparison result to the user. The experimental results show that this scheme can effectively overcome the storage and misjudgment problems caused by existing blockchain-based certificate revocation schemes and improve the query efficiency of certificate revocation information.

## 1. Introduction

Public key infrastructure (PKI) collects hardware, software, people, policies, and procedures. It can realize the generation, management, storage, distribution, and revocation of keys and certificates based on public key cryptosystems [1], which is the foundation and core of network security construction. It is now widely used in secure e-mail, virtual private networks, e-commerce, and e-government and is the basis for achieving network security [2]. Certificate revocation is one of the core functions of PKI, which indicates the end of certificate life. When personal identity information changes or the private key of the certificate is leaked, or the fraudulent behavior of the certificate owner, the certificate user should promptly submit a certificate revocation request to CA. CA should also put certificates into the publicly released Certificate Revocation Lists (CRLs) in time. The traditional certificate revocation method is to store revoked certificate number in the LDAP directory server, and the user is informed of the certificate revocation information by querying LDAP. This centralized query approach suffers from the trustworthiness of the LDAP administrators, and the LDAP directory server can become a performance bottleneck as the number of accesses increases.

Blockchain [3, 4] has been developed relatively quickly in recent years. In essence, it is a shared database, a distributed ledger technology based on the point-to-point network, providing a set of distributed data structures, interaction mechanisms, and computing paradigms [5], with decentralized storage, decentralization, tamper-proof and traceable characteristics [6–8]. Blockchain has laid a solid foundation of "trust." It is widely used in data security [9], becoming a better solution to the problem of traditional certificate revocation due to its superiority in transparency, traceability, and security [10]. There are many research results. Fromknecht et al. [11] designed a fully decentralized PKI using consistency provided by the Namecoin blockchain. Kubilay et al. [12] proposed a new blockchain-based

PKI architecture for certificate transparency. In particular, Rabieh et al. [13] used Bloom filters to reduce the size of CRL, and Medury et al. [14] and Huang et al. [15] used cuckoo filters to quickly verify revoked certificates and use blockchain publishing filters. However, existing blockchain-based certificate revocation schemes also have two problems: first, the storage overhead on the blockchain is relatively large. For example [13] uses the insertion of certificate fingerprints in Bloom filters. When a new certificate revocation transaction is generated, it is necessary to publish a new complete filter to the blockchain, with each block storing a complete array. Second, as the number of revoked certificates increases, the false-positive rate of certificate status increases accordingly. The Bloom filter used in [13] is a probabilistic data structure designed by multiple hash function algorithms. The principle is to calculate the hash value by hash function and then map this value to an array set to 1. However, different values may generate the same address, resulting in a hash collision, causing the problem that filter query results do not match actual data. The cuckoo filter used in [15] is a probabilistic data structure designed based on the Cuckoo Hashing algorithm. The principle is to calculate the fingerprint and hash value of the data and calculate another hash value from the fingerprint and hash value. It maps two hash values to two locations. If the insertion fails at both positions, one fingerprint is randomly squeezed out, and a new position is found for that fingerprint again. This method may cause a hash collision on fingerprint information in extreme cases, leading to the misjudgment that elements outside the set exist in the set. Benaloh and Mare [16] first proposed accumulators [17]. It can hash a large set of inputs into a short value. Moreover, given an accumulator, an element, and a membership witness, it can verify the existence of the element in the cumulative set [11]. Member witnesses are generated when relevant elements are added to the accumulator and are usually updated when the collection is changed. Member witnesses that are not elements of the accumulator are difficult to find computationally. The feature of the accumulator is that when an element is added or removed, accumulated value and membership proofs can be effectively updated, and it supports member proofs and nonmember proofs. Accumulator uses the strong RSA assumption in cryptography to ensure security and zero-knowledge proofs. It ensured that users do not reveal information about themselves when proving their legitimacy to the verifier.

Therefore, this article proposes a public key infrastructure certificate revocation scheme based on blockchain and accumulator. First, we expand the certificate structure and add a new revocation factor and blockchain access information to the certificate extension, which is populated by CA invoking smart contracts when generating certificates. Then, when a certificate is to be revoked, CA generates a revocation fingerprint through the revocation factor and accumulator and publishes it to the blockchain. Finally, when verifying certificate status, CA verifies the validity of the certificate according to the revocation factor and revocation fingerprint of the blockchain. The experimental results show that this scheme can effectively overcome storage and misjudgment problems caused by the previous certificate revocation scheme and improve the query efficiency of certificate revocation information. The rest of this article is organized as follows: Section 2 describes current research work on certificate revocation. Section 3 describes the essential concepts of this system. Section 4 describes the system design. Then, Section 5 presents a feature analysis of this article. Section 6 describes experiments and gives a performance evaluation of the proposed method. Section 7 concludes the whole article and presents future research prospects.

## 2. Related Work

For certificate revocation of PKI, many research results have been achieved. The following will analyze and summarize the current primary certificate revocation mechanisms and blockchain-based certificate revocation mechanisms.

*2.1. Main Certificate Revocation Schemes.* Certificate Revocation List (CRL) [18, 19] is a time-stamped list in which all certificate information that has been revoked or hung is listed, issued by the certification authority CA and published periodically. CRL contains two fields: the current update date and the next update date. Users can determine whether the current CRL is the latest from two date information, and CRL contains the signature of CA. So CRL can be stored in any node on the network. To check the validity of a certificate, the verifier initiates a request to the LDAP directory server hosting the corresponding CRL with the CA identifier parameters that issued the certificate. Then it receives the latest CRL generated by CA and checks the CRL signature and its validity. Finally, the certificate is searched in CRL to determine whether the certificate and key pair are trusted. The advantages of the traditional CRL approach are simplicity, information richness, and low risk. The disadvantages are high bandwidth cost, low query efficiency, and long delay time. The size of CRL is its main drawback. The amount of communication between user and directory server is heavy. Each verification of the public key certificate requires downloading the entire CRL, which requires high bandwidth for verification and update. When the scale of CA becomes larger and larger and users use certificate information more and more frequently, a large number of users download new CRLs on LDAP. CAs have to keep publishing new CRLs to LDAP, which at this time tends to cause congestion within CRL requests. The feature greatly limits the scalability of this method. At present, many improved CRL schemes have been proposed, and some well-known ones are described below. Incremental distribution (Delta-CRL) [20] provides a more efficient way to distribute certificate status information. Instead of generating a complete and potentially growing CRL every time a certificate is revoked, the list only records all the unexpired certificates that have been revoked since the last CRL was issued. Clients do not have to download the entire CRL but only maintain their own CRL database and keep it updated with a Delta-CRL that is much smaller than the size of the entire CRL, saving

communication bandwidth and time. Delta-CRL aims to solve the scalability problem of downloading CRLs. However, Delta-CRL only represents a part of CRL, and revocation information can only be used after it is associated with the main CRL. That is, any request issued at a certain point in time requires a complete CRL. This scheme cannot solve the problems of verification time and computational complexity of revoked certificates. CRL distribution points (CRL-DP) [21] is a way for CA to address scalability by partitioning CRL using CRL distribution points on a compromise and routine revocation basis. The main idea is that system divides the entire authentication space into small fragments according to some classification. Each fragment is associated with a particular CRL distribution point, which can be located on a different host or on a different directory on the same host. Clients checking certificate status can access the CRL distribution point specified in the certificate instead of the CRL distribution point in the main CRL. Therefore, the CRL distribution point reduces the length of CRL downloaded by the user, which is more advantageous than complete CRL in balancing network load and improving authentication efficiency. However, this method does not reduce peak requests and increases users' average request rate and waiting time when they need to query multiple segments [1]. Moreover, since the location of CRL distribution points is fixed throughout the life of the certificate, CA must know in advance how to segment the CRL information and fix the location of CRL distribution points, which is also a problem of the method. Redirect certificate revocation list (RCRL) [22] can solve the problem that the location in the CRL release point cannot be changed. In this mechanism, a new critical CRL extension is defined. This extension consists of a range that covers authenticated certificates and a pointer to the new CRL location of the problematic certificate. Even though redirected CRL solves the problem of fixed distribution point locations. It still causes an increase in the average CRL request rate and longer user wait times as the number of CRL segments increases. Indirect CRL [22] enables the publication of revocation information from multiple CAs in a single CRL. That is, multiple CAs can use the same CRL distribution point. The use of indirect CRLs reduces the total number of CRLs that users need to retrieve during the certificate validation process, reducing traffic load and cost. However, since revocation information comes from different places, it is necessary to determine the CA of each item in the certificate revocation list. Therefore, a certificate issuer field needs to be set in each item. The distribution point is maintained by another trusted third party, increasing the difficulty of maintaining a single distribution point. Another alternative to RL is the Certificate Revocation Status (CRS) [23, 24], which is an authentication dictionary data structure with evidence having the characteristic of being delivered through unscientific third parties [22]. CRS was designed in accordance with the following principles: increasing the amount of communication between CA and directory during the update of revocation information and being able to minimize the length of evidence obtained when a user queries the status of the certificate from a directory (this contains all the information of revoked certificate in CRL). CA sends a signed statement to the CRS directory every day stating the status of individual issued certificates, and each unexpired certificate has a signed statement. When a user queries for certificate revocation status, CRS Directory replies with information that the user can use to verify the requested status. CRS reduces the communication load between server and end entity, achieving an overall performance improvement compared to the CRL method. However, it greatly increases the communication load between the server and CA.

In addition to this, an alternative to the CRL scheme is Certificate Revocation Tree (CRT) [25]. It is usually a Merkle hash tree representing all certificate revocation information for a given PKI domain, providing a set of statements about the certificate sequence numbers in leaves. The main advantage of this approach is that we do not need a complete CRL to provide certificate validation. However, its main disadvantage is updating, since any change in the revocation certificate set may cause the entire list to be recomputed, resulting in a continuous workload [26].

Online Certificate Status Protocol (OCSP) [27] is an online revocation system that relies on a request/response mechanism. It acts between the client and the server and provides a way for applications to obtain certificate status online. The client, called an OCSP requester, generates an OCSP request to send to the server if it wants to verify the status of one or more certificates. The server, called an OCSP responder, first verifies the request's syntax and semantics after receiving the client's request and then constructs an OCSP response to return to the requester. Revocation information is obtained at the server of the OCSP responder, which receives it directly from CA. In fact, the CA does not sign the OCSP response, so the revocation server must be trusted by the CA. OCSP approach solves low timeliness and revocation information update problems. However, this method has some drawbacks, mainly (1) since this method is centralized, the OCSP server represents a single point of failure [28]. (2) OCSP responds to verify the certificate's revocation status without checking the validity sequence number. A malicious user can use the validation flood server to request a certificate not belonging to CA. This makes the server work concentrated will lead to denial of service. (3) OCSP lookup has a high overhead [29, 30]. (4) OCSP is an ineffective online scheme for offline systems [30]. (5) OCSP can provide real-time responses to revocation queries, but it is unclear whether these responses contain updated revocation information. (6) OCSP approaches introduce privacy risks. OCSP responders know which certificates end users are verifying, so they can track which sites users are visiting [30].

*2.2. Blockchain-Based Revocation Schemes.* In recent years, blockchain has become popular in certificate revocation research. Blockchain-based technologies are appealing because they allow for secure, robust, and trustworthy solutions and bring improvements compared to current

technologies or management systems in terms of transparency and traceability. It is the ideal technology for PKI design and deployment [31]. The following describes several blockchain-based PKI methods, focusing on their certificate revocation management component.

Fromknecht et al. [11] proposed a fully decentralized PKI that leverages the consistency provided by the Namecoin blockchain to provide strong identity retention guarantees, which [11] has five functions: register, update, find, verify, and withdraw. Although Fromknecht et al. [11] solved some problems, the method still has many shortcomings. As in the high cost of mining and public key lookup and verification, there is no actual verification of the linkability of ID links to registered public keys. Moreover, during the revocation process of Fromknecht et al. [11], the owner of the identity ID can revoke its public key only by publishing a transaction to the blockchain. The entire revocation process is completely handled by the owner himself, which will cause many problems; for example, (1) handling revocation by the user himself is a difficult task as it requires some expertise. Furthermore, a user cannot know if the key has been compromised. (2) Malicious users will not revoke their keys. (3) To verify the certificate's status, the scheme must first verify that a revoked certificate is published in the blockchain. It is all about browsing the blockchain to ensure that the certificate has not been revoked. However, censoring search content in blockchain can take much time. Hu et al.[32] proposed Certificate Revocation Guard (CRG), which intercepts all TLS communications from entities such as organizational gateways using an intermediate box that performs OCSP requests to check certificate revocation status. If a revoked certificate is detected, a malformed certificate is returned to the client, effectively blocking the connection. The policy does not require any modification by clients to participate. However, mobile clients such as laptops and smartphones will lose protection when they leave the network due to using intermediaries. Hewa et al. [33] proposed the application of an Elliptic Curve Qu-Vanstone (ECQV) certificate, which is lightweight for resource-constrained IoT devices. Additionally, they integrate blockchain-based smart contracts to handle certificate-related operations. They apply smart contracts to certificate issuance and develop a smart contract-based threat scoring mechanism to revoke certificates automatically. The lightweight nature of ECQV certificates enables distributed ledgers to store, renew, and revoke certificates. Kubilay et al. [12] proposed a new blockchain-based certificate transparency PKI architecture, called CertLedger, and provided an ideal certificate revocation transparency. The revocation status of all TLS certificates, the entire revocation process, and trusted CA management is carried out in CertLedger. BARS [34] is a blockchain-based anonymous reputation system to break the linkability between real identities and public keys to preserve privacy. BARS has two main contributions: first, they exploit the features of blockchain to extend conventional public key infrastructure with an effective privacy-preserving authentication mechanism. The linkability between the public key and the real identity of a vehicle is eliminated when a certificate authority (CA) operates the certificate issuance and revocation. Second, the algorithm evaluates the trustworthiness of each vehicle according to the authenticity of broadcasted messages and opinions from other vehicles. All the messages are recorded on the blockchain. The reputation score provides an incentive for internal vehicles to prevent misbehavior and mitigate forged messages' distribution. In [35], Malik et al. proposed a framework for transaction authentication and revocation that authenticates vehicles and speedily updates revoked vehicles' status in the shared blockchain ledger with the PoA mechanism. This method reduces the dependency on CA in the validation process. Feng et al. [36] presented an efficient privacy-preserving authentication model called EPAM that shortens the time of checking Certificate Revocation Lists (CRLs), alleviates the presentation problem during mutual authentication, and achieves privacy properties such as anonymity and unlinkability. Wang et al. [37] utilized a smart contract as a transparent agent to manage the revocations. The user sends a revocation request to the smart contract, and the smart contract periodically transmits valid requests to CA. That scheme directly displays the revocation identity to a smart contract, which may violate the user's privacy and face scalability issues. Lin et al. [38] proposed a novel BCPPA protocol. The Elliptic Curve Digital Signature Algorithm (ECDSA) based on PKI is used in this scheme. The algorithm is based on a public blockchain (Ethereum) for secure communication. Participating vehicles do not need to store "private keys," further reducing verification time and costs. Yao et al. [39] proposed a privacy-preserving blockchain-based certificate status validation scheme called PBCert. The scheme is designed to store all revoked certificates in the OCSP server, and only the minimal control information (namely, certificate hashes and related operation block height) is stored in the blockchain. The scheme uses bloom filters to improve the efficiency of client-side status validation. Rabieh et al. [13], for the scalability of Advanced Metering Infrastructure (AMI) networks, partitioned the network into clusters of SMs. However, there is a trade-off between the overhead of a certificate authority (CA) and the overhead of a cluster. Bloom filters are used to reduce the size of CRL. However, bloom filters will give false positives. The additional distribution of the list of certificates that trigger false positives through the gateway and CA identifies and eliminates false positives, but this adds overhead. Medury et al. [14] and Huang et al. [15] used the cuckoo filter to verify revoked certificates quickly, and they stored certificate information and cuckoo filter coefficients in the blockchain. This method can reduce the cost of certificate storage, but there is a problem of false-positive rate caused by the filter.

To sum up, although many research results have been achieved in certificate revocation. However, there are still two problems in the combination of blockchain and certificate revocation: storage overhead on the blockchain is relatively large. As the number of revoked certificates

increases, the rate of misclassification of certificate status will increase accordingly. These two issues are still not better addressed.

## 3. Related Algorithms of the Accumulator

In this article, the accumulator is used for revocation factors and status verification. We introduce the related algorithms of cryptographic accumulators as follows:

(1) KeyGen $(k, M)$ is a probabilistic algorithm that is executed in order to instantiate the scheme. It takes as input a security parameter $1^k$ and the upper bound $M$ on the number of accumulated elements and returns an accumulator parameter $P = (P_u, P_r)$, where $P_u$ is a public key and $P_r$ is a private key.

(2) AccVal $(L, P)$ is a probabilistic algorithm that computes an accumulated value. It takes as input a set of elements $L = \{C_1, C_2, C_3, \ldots, \ldots, C_m\}(1 < m \le M)$ and returns an accumulated value $v$, along with some additional information $a_c$ and $A_l$.

(3) WitGen $(a_c, A_l, P_u)$ is a probabilistic algorithm that creates the witness for every element. It takes as input the auxiliary information $a_c$ and $A_l$ and the parameter $P$ and returns a witness $W_i$ for each $C_i(i = 1, 2, \ldots, m)$.

(4) Verify $(c, W, v, P_u)$ is a deterministic algorithm that verifies that a given element is accumulated in the value $v$. It takes as input an element $c$, its witness $W$, the accumulated value $v$, and the public key $P_u$ and returns YES if the witness $W$ constitutes a valid proof that $c$ has been accumulated in $v$, or NO otherwise.

(5) AddEle $(L^-, a_c, v, P)$ is a probabilistic algorithm that adds new elements to the accumulator and generates a new accumulated value. It takes as input a set of new elements $L^+ = \{c_1^+, c_2^+, \ldots, c_i^+\}\{L^+ \subset C, 1 \le i \le M - m\}$, auxiliary information $a_c$, the accumulated value $v$, and the parameter $P$, returns a new accumulated value $v'$ corresponding to the set $L^+ \cup L$, witnesses $\{W_1^+, L, W_k^+\}$ for the newly inserted elements $\{c_1^+, c_2^+, \ldots, c_k^+\}$, along with new auxiliary information $a_c$ and $a_u$.

(6) DelEle $(L^-, a_c, v, P)$ is a probabilistic algorithm that deletes some elements from the accumulated value. It takes as input a set of elements $L^-\{c_1^-, c_2^-, \ldots, c_k^-\}(L^- \subset L, 1 \le k \le m)$ that are to be deleted, the auxiliary information $a_c$, the accumulated value $v$, and the parameter $P$ and returns a new accumulated value $v'$ corresponding to the set $L/L^-$, along with new auxiliary information $a_c$ and $a_u$.

(7) UpdateWit $(W_i, a_u, p_u)$ is a deterministic algorithm that updates witness for the elements that have been accumulated in $v$ and $v'$ after adding or deleting operations to the set $L$. It takes as input the witness $W_i$, the auxiliary information $a_u$, and the public key $P_u$ and returns an updated witness $W_i'$, proving that the element $c_i$ is accumulated in the new value $v'$.

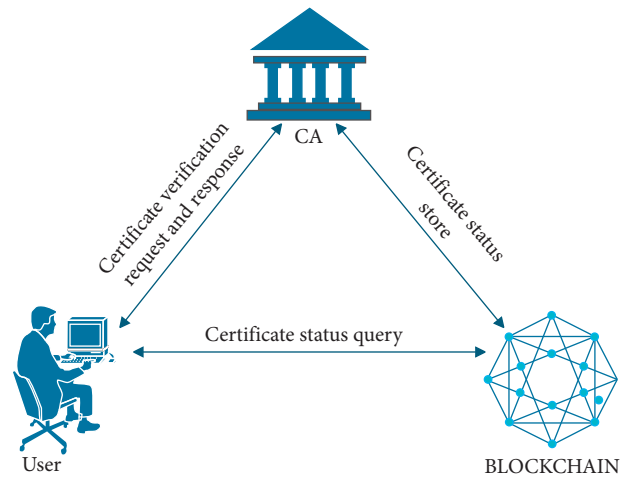


Figure 1: System structure.

(8) Input $a_c$, Al, and parameter $P$ and output the witness $W_i(i = 1, 2, \ldots, m)$ for each $C_i(i = 1, 2, \ldots, m)$.

## 4. Certificate Revocation Scheme Based on Blockchain and Accumulator

Verification of certificate revocation status is a critical link in the reliability of public key infrastructure systems. This article's method aims for the reliable distribution and storage of certificate revocation information and designs a public key infrastructure certificate revocation scheme based on blockchain and accumulator. The core is to introduce a field to expand the X.509 certificate structure, namely, the revocation factor. The revocation factor is issued by the blockchain and embedded in the revocation certificate by CA. Blockchain stores the accumulator value. Whenever CA revokes a certificate, it recalculates the corresponding accumulator value and provides a new transaction to put it stored in the blockchain. Since this system only needs to detect revocation information, it only uses the accumulator's accumulation and nonmember certification functions. Newly generated certificates do not need to broadcast new accumulator values. Only each revocation needs to broadcast accumulator values, thus detecting whether certificates have been revoked. When the user checks whether a certificate is revoked, the smart contract is used to verify whether the revocation factor is the factor of the blockchain accumulator value. Finally, get the certificate status.

The whole system includes three entities: certificate authority, blockchain, and user, as shown in Figure 1:

(1) Certificate Authority (CA): CA is an entity that revokes a certificate. CA responds to the user's certificate request and performs certificate issuance. CA sends a new transaction to the blockchain for each certificate revocation to share the information.

(2) Blockchain: A distributed ledger stores revocation information, storing accumulator values on the blockchain.

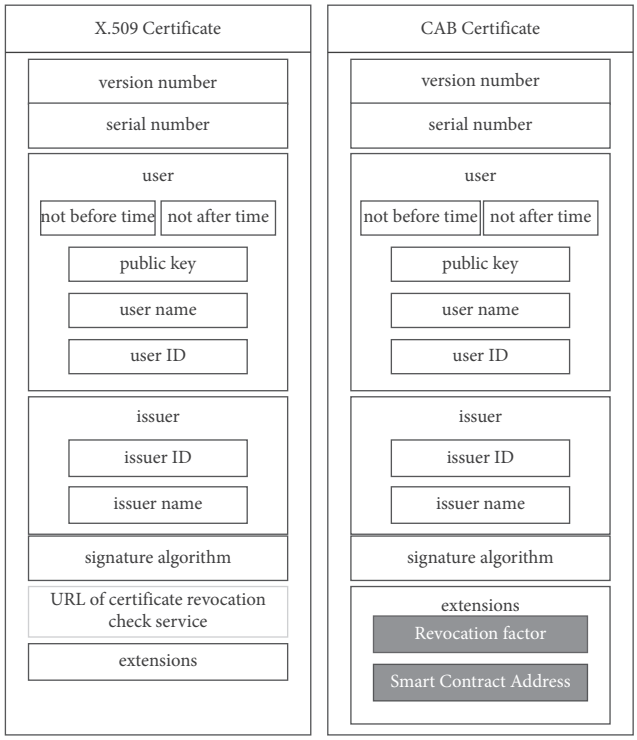


Figure 2: X.509 certificate and CAB certificate.

(3) User: An entity that requests and receives certificates issued by CA. The user submits identity information in the registration stage, and the communication requires identity verification using a digital certificate. The authentication step includes revocation status verification to ensure a valid certificate status before establishing a communication connection.

*4.1. Certificate Structure Design.* This article designs a new certificate based on the X.509 certificate. X.509 certificate and CAB certificate are shown in Figure 2.

Compared with the traditional X.509 certificate, the main improvements are as follows:

(1) This article adds a revocation factor to the extension. When CA generates a certificate for the user, the smart contract invoking the licensed blockchain uses the accumulator to generate a revocation factor and an accumulator value. The revocation factor is returned to CA, the certificate is inserted, and the accumulator value is written into the blockchain. CA uses the revocation factor to verify that the certificate is in the accumulator when the client verifies certificate status.

(2) The certificate designed in this article changes the URL of the certificate revocation check service to the smart contract address. When traditional PKI queries whether a certificate is revoked, it finds the location of the CRL distribution point according to the URL of the certificate revocation check service. It downloads the CRL list to check the certificate serial number in it to check the certificate status. The scheme changes the URL module to the address of the smart contract. When a client needs to query the certificate status, it only needs to verify whether the unique value contained in the certificate is included in the accumulator value according to the revocation factor provided by the user.

When the customer applies for a certificate, the information is passed to the verification center in this system. After the verification center verifies the customer information, CA issues a certificate as follows:

```
Certificate : = SEQUENCE {
        tbsCertificate TBSCertificate, signatureAlgorithm AlgorithmIdentifier,
        signatureValue BITSTRING
}
        TBSCertificate : = SEQUENCE {
        version v3, -- Certificate version number
        serialNumberCertificateSerialNumber default;--Serial number
        signatureAlgorithmIdentifier default, --Signature algorithm identification
        issuerName default,--issuer name
        validity default, --Certificate validity period
        subjectName CAB-Certification, --Certificate subject Name
        subjectPublicKeyInfo, SubjectPublicKeyInfo,--Certificate public key
        issuerUniqueID default, --Certificate issuer ID
        subjectUniqueID default,--Certificate subject ID
        extensions Extension--Extension
}
        Extension : = SEQUENCE {
        extnID OBJECT IDENTIFIER,
        critical Boolean DEFAULT FALSE,
        extnValue OCTET STRING,
        UndoRF default, --Revocation search factor
        CPSDistributionPoints default, --Revocation check address
}
```

The certificate is designed based on X.509 certificate. The certificate carries out regular authentication but differs from the CRL mechanism in the state check part. It provides a witness of the unique value contained in the certificate to make the client believe that the certificate is still valid. Therefore, the revocation factor provided by the blockchain is added to the extension.

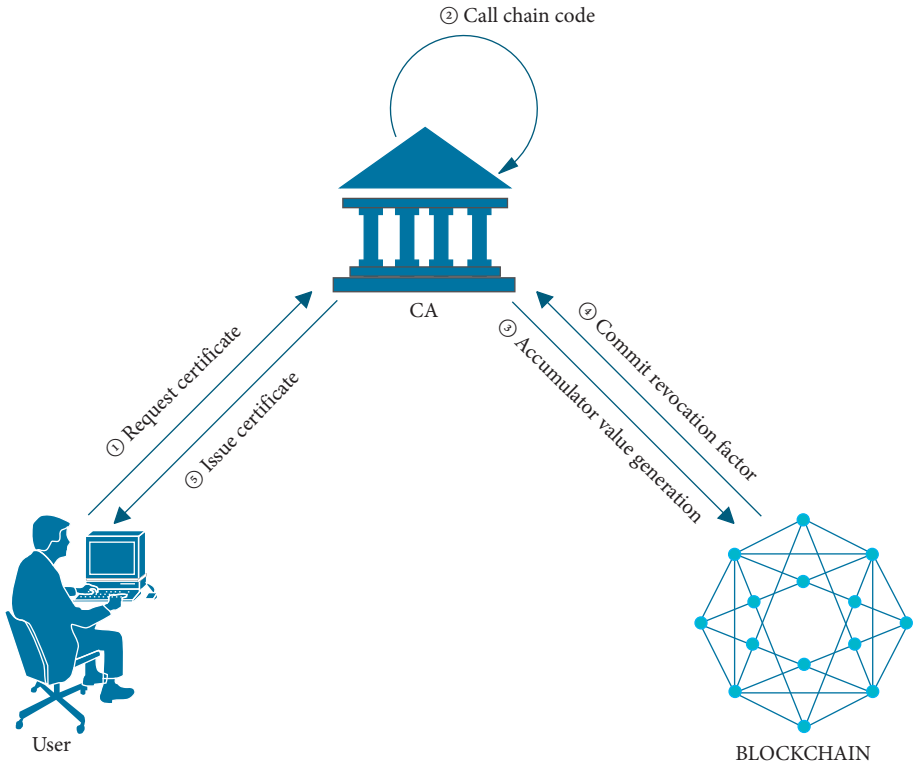


Figure 3: Revocation factor generation.

*4.2. Accumulator-Based Revocation Factor Generation.* The user initiates a certificate request to CA. CA makes the user information, the public key obtained by the KMC, and the revocation factor generated by calling the smart contract into a certificate. The revocation factor is generated by the accumulator distributed in the blockchain. Accumulator performs the following steps to calculate the accumulated value and the revocation factor: first, it concatenates the certificate's serial number with its issuer's public key to obtain a unique string that prevents problems caused by having the same serial number from different CAs. It computes a relative prime number from a string. Then, add this prime number to the accumulation list through proveMembership, and calculate a newly accumulated value accValue and the corresponding revocation factor witness, as shown in Figure 3.

A user initiates a certificate revocation request to the CA, which issues a certificate to the user containing a revocation factor. The revocation factor is generated by the blockchain distribution accumulator, which aims to use blockchain to improve the availability of revocation information and reduce the risk of insider threats caused by compromised nodes in the distribution system. The certificate revocation process is shown in Figure 4.

(1) User ⟶ CA: req ()

The user initiates a certificate request to CA.

(2) CA: verify (certID, $key_{pub}$)

After receiving the request, CA assigns a unique certificate to the current operation initiator. The process of calling the accumulator in the smart contract and generating the accumulator value and revocation factor is as follows.

(a) Add (certID): Generate a new accumulator value(accValue′) by passing in the object and the current certificate accumulator value (accvalue).
(b) WitCreate (certID, accValue′): The corresponding revocation factor (witness) is generated through a public key ($key_{pub}$), accumulated value (accValue), and element (certID).
(c) updateAcc (data): Write the updated accumulator value to the blockchain.

(3) CA⟶User: res (certID)

The system returns the certificate to the user. The specific algorithm is Algorithm 1.

The input parameters of Algorithm 1: accValue is the certificate accumulator value on the blockchain, member is the certificate member to be added to the accumulator, and key is the user key. The output parameters: accValue′ is the updated accumulator value, and witness is the revocation factor, which is the generated member witness. The function of lines 1 to 3 is to get the accumulator value on the blockchain corresponding to the issuer. The function of line 4 is to use verify () function to verify that the member is in that accumulator value. The function of line 5 is to add this member to the accumulator if it is not in this accumulator. The function of line 6 is to generate a new accumulator value accValue′ and a new revocation factor witness. The function of line 7 is to store the accumulator

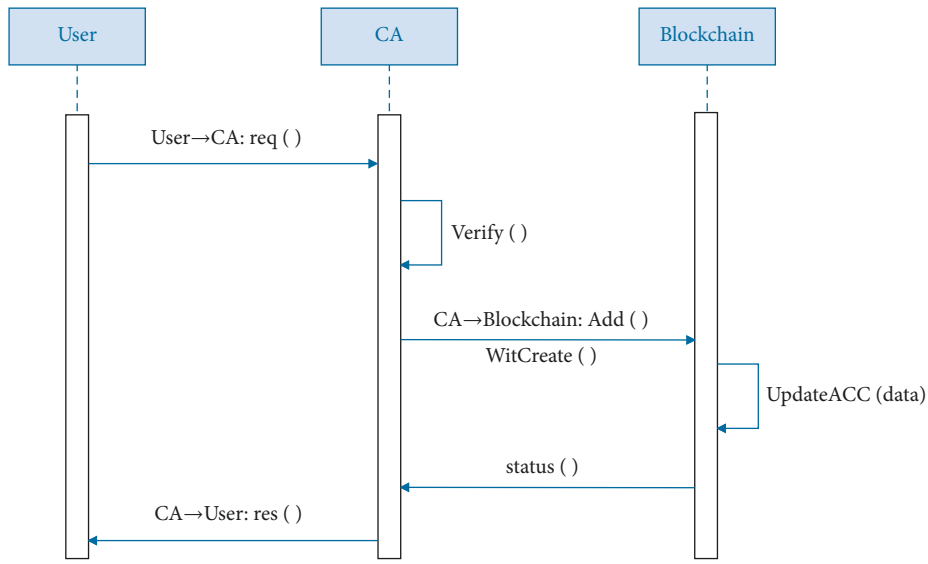


Figure 4: Certificate revocation process.

value in the blockchain. The function of line 8 is to return the revocation factor to CA. Instead, tell CA that this member already exists.

*4.3. Accumulator-Based Revocation Factor Update.* When authenticating, users use revocation factors to prove the validity of their identity. The revocation list is compressed into a short value using an accumulator to verify the certificate's validity. This short value can be easily updated and distributed on a properly instantiated and managed blockchain network. When a certificate is added to the revocation accumulator, both the accumulator value and revocation factor are updated. The process for updating the revocation factor is shown in Figure 5.

(1) User ⟶ CA: req(certID, witness, sign (certID, witness)).

As shown in Figure 6, the user first initiates a request to revoke the certificate to CA, where certID is the unique identity certificate of this user, witness is the revocation factor issued by blockchain for this certificate, and sign (certID, witness) represents the user's signature value for their account and revocation factor.

(2) CA: verify (certID, witness, sign (certID, witness))

After CA receives the request, it performs the verification signature operation to ensure that the certificate corresponding to the current operation initiator belongs to the current user. After verification is passed, the system calls the revocation certificate function in the smart contract. The revocation protocol process is as follows.

(a) query (certID): the deserialized accumulator object verifies if the certificate is in the blockchain by passing in the revocation factor and current certificate accumulator value.

(b) revokeFromAcc (certID): call revoke certificate interface of the accumulator to remove the member from the accumulator and recalculate the accumulator value.

(c) updateAcc (data): write updated accumulator value to the blockchain.

(3) CA ⟶ User: res(certID, status)

The system returns the certificate certID and status of the revoked certificate to the user, where status contains revocation success or revocation failure. The algorithm is shown as follows.

When verification credential needs to be regenerated, execute MemWitUp algorithm to update accumulator value and revocation factor. The specific algorithm is Algorithm 2.

The description of Algorithm 2 is as follows. The function of lines 1 to 3 is to get the accumulator value on the blockchain corresponding to the issuer. The function of lines 4 to 6 is to use verify () function to verify that the member is in that accumulator value. The function of lines 7 to 8 is to delete the member using the delete () function if it is included in the accumulator value. The function of line 9 is to generate a new accumulator value and revocation factor using the proveMembership () function. The function of lines 10 to 11 stores the accumulator value in the blockchain and returns the revocation factor to CA. The function of lines 12 to 13 is that the member is not in the accumulator and cannot be updated.

*4.4. Accumulator-Based Certificate Status Verification.* The user applies for certificate status query operation and submits the user's certificate ID, revocation factor, and signature parameters. After CA receives the request, it performs the verification signature operation to ensure that the certificate corresponding to the current operation initiator belongs to the current user. After

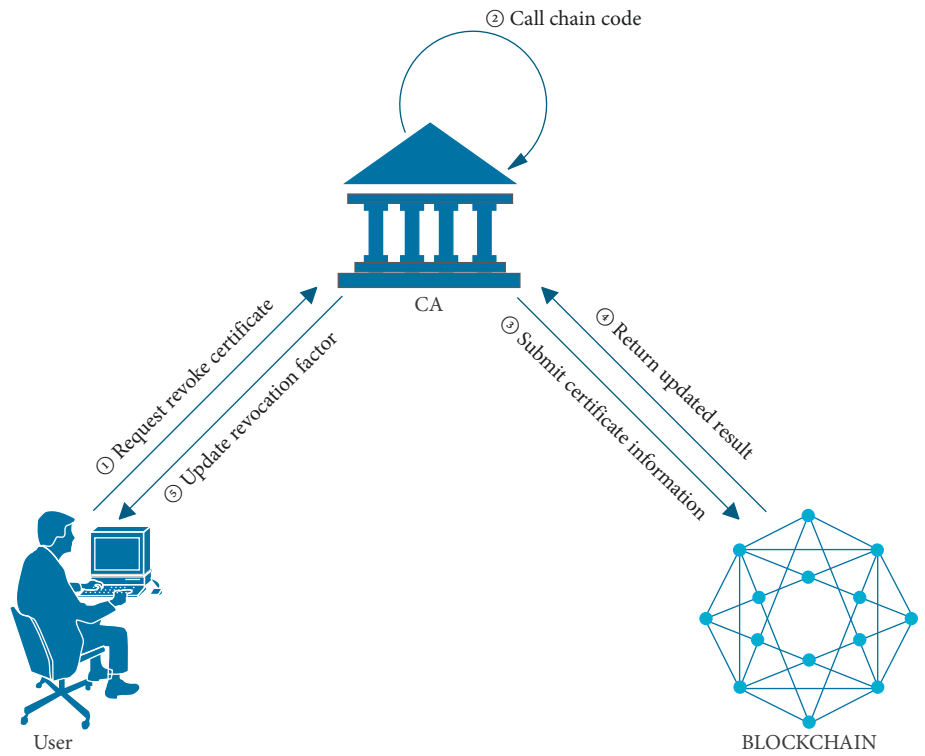


Figure 5: Updating revocation factor process.

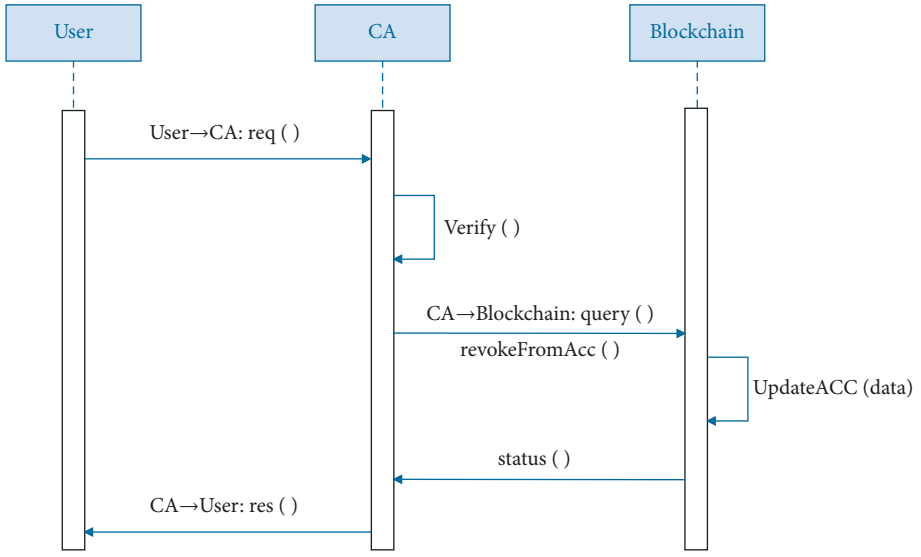


Figure 6: Updating accumulator value process.

verification is passed, the revocation status check function in the smart contract is called, and the queried certificate status is returned to the user by executing the member verification algorithm. The specific process is shown in Figure 7.

A certificate query refers to querying the corresponding certificate status from the blockchain through the information given by the user. Protocol design for querying certificate revocation status is shown in Figure 8.

(1) User ⟶ CA: queryCert (certID, witness, sign ())

The user applies for certificate status query operation, and submitted parameters represent the user's certID, the revocation factor, and the signature value.

(2) CA: verify (certID, witness, sign ())

After CA receives the request, it performs the verification signature operation to ensure that the

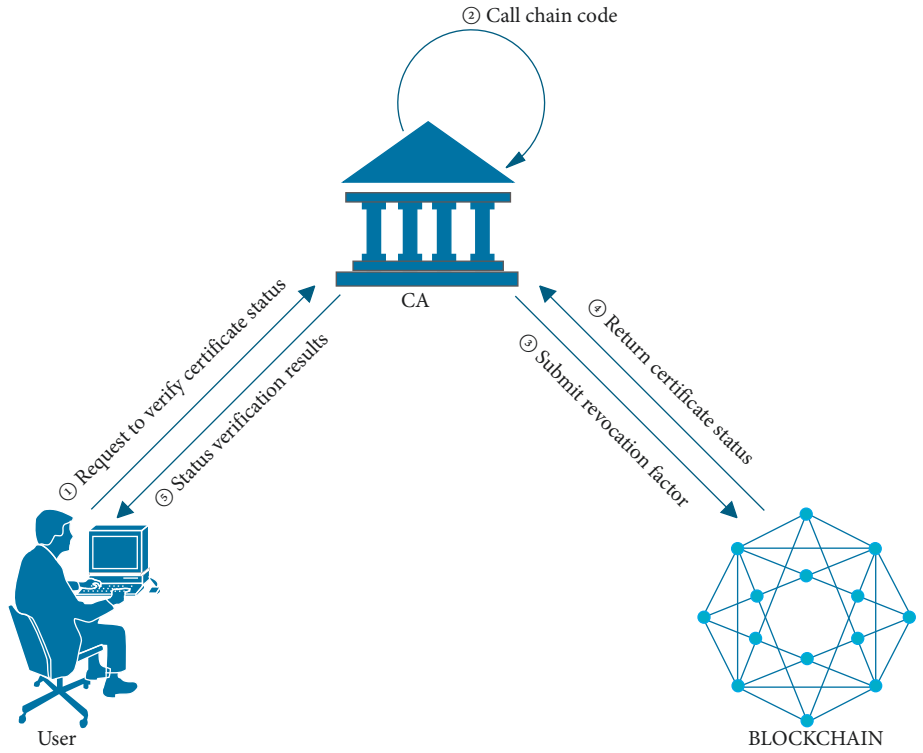


Figure 7: Revocation status verification.

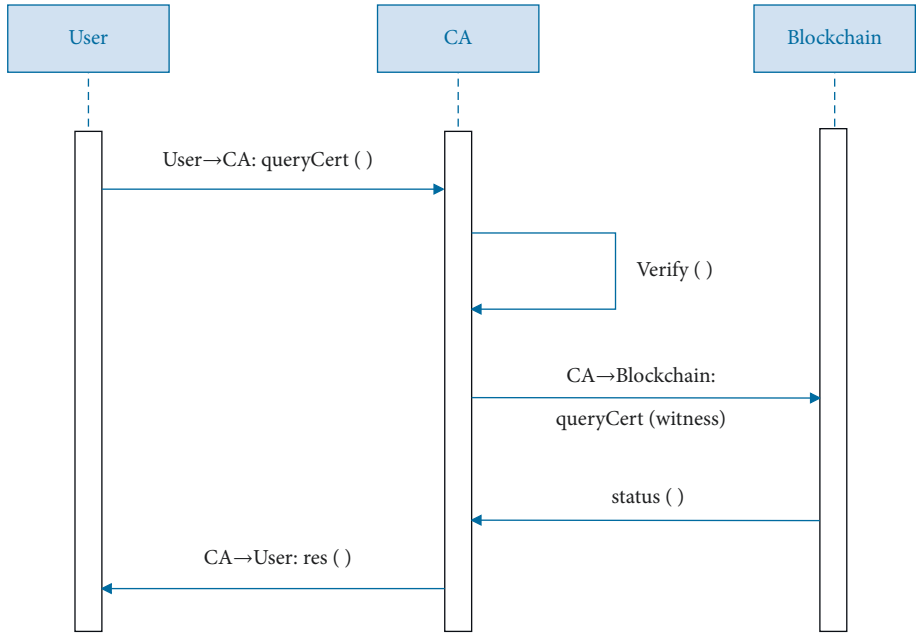


Figure 8: Query certificate status process.

certificate corresponding to the current operation initiator belongs to the current user. After the verification is passed, the revocation status check function in the smart contract is called. The protocol flow is as follows.

(a) queryCert (certID): By executing the member verification algorithm in the accumulator, verify $\text{witness}_{c_i}^{c_i} \text{mod} N = \text{acc}_{\text{cert}}$ correctness and return the queried certificate status to the user.

(b) CA ⟶ User: {data} returns the certificate status verified from the blockchain.

The specific algorithm is Algorithm 3.

The description of Algorithm 3 is as follows. The function of lines 1 to 4 is to get the accumulator value on the

```
Input: accValue, member, key
Output: accValue′, witness
(1) ChaincodeStub stub = ctx.getStub ();
(2) byte[] ojectBytes = stub.getState (Accumulator.class.getSimpleName ());
(3) Accumulator accValue = deserialize(ojectBytes); //deserialize the accumulator object
(4) result ← Acc.verify (member); //verify that current certificate exists
(5) accValue ← Acc.add(member); //add member to accumulator
(6) witness = acc.proveMembership (sha256 (cert, n)); //compute new accumulator value and new revocation factor
(7) SendBlockchainTransaction (accValue′); //update accValue to the blockchain
(8) return accValue′, witness; //return revocation factor, and the accumulator value is saved to the blockchain
```

Algorithm 1: Generative algorithm, provemembership.

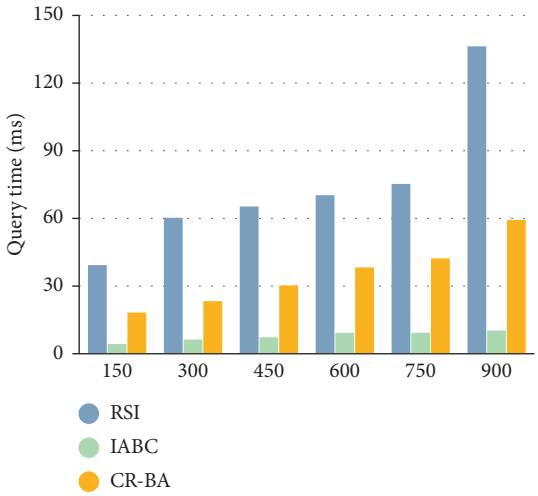


Figure 9: Query time of the revoked certificate.

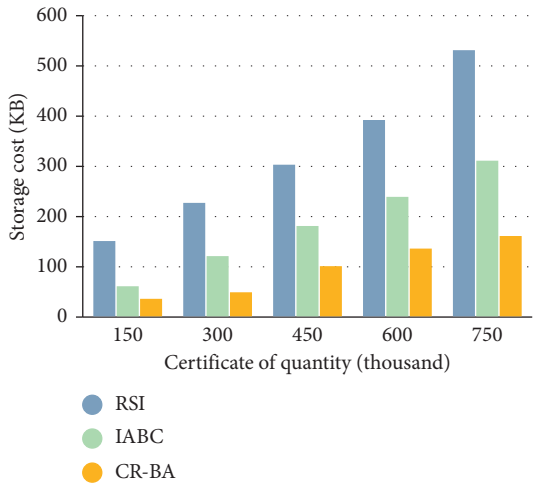


Figure 10: Certificate storage cost comparison.

blockchain corresponding to the issuer. The function of line 5 is to use the verify () function to verify that the member is in that accumulator value. The function of lines 6 to 7 is to inform CA that this certificate is still valid if this member is in the accumulator. The function of lines 7 to 8 is to return the certificate status to CA that it has been revoked.

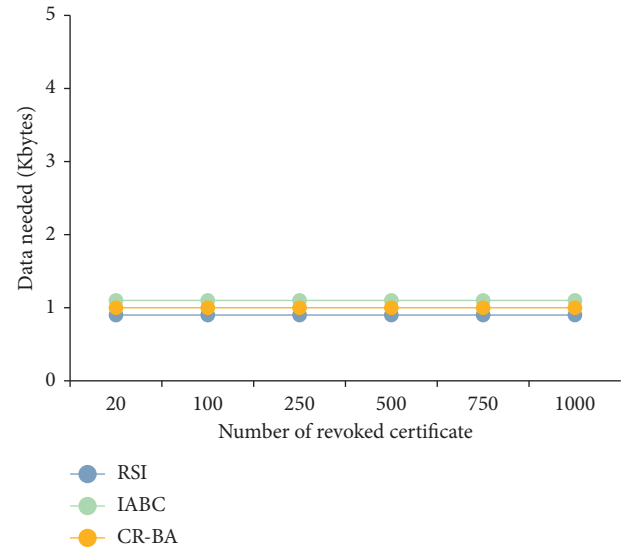


Figure 11: Amount of data needed to exchange to provide a response on the revocation status of a nonrevoked certificate.

As a security property of the accumulators, the probability of finding a nonmember witness for an element in accumulating set is negligible. Therefore, in the case of certificate revocation, the server cannot update its witness and prove it is not on the revocation list.

## 5. Feature Analysis

*5.1. Security.* The system uses accumulators and blockchain. We use an accumulator to compare the revocation list into a digest, which is updated and distributed through a properly instantiated and managed blockchain network. This small digest allows us to easily distribute validation data, reduce communication overhead and improve system scalability. The accumulator in this article is a secure accumulator based on a strong RSA assumption. Under this assumption, the problem of finding $f(w, m) = w^m \bmod n$ that satisfies the condition is polynomials hard to solve in a short time. Given $v$ and $m$, finding a $w$ such that $v = f(x, y)$ is difficult, so the accumulator $f(w, m) = w^m \bmod n$ is secure. In this article, we use blockchain to distribute accumulator values and blockchain to improve the availability of revocation information and reduce the risk of

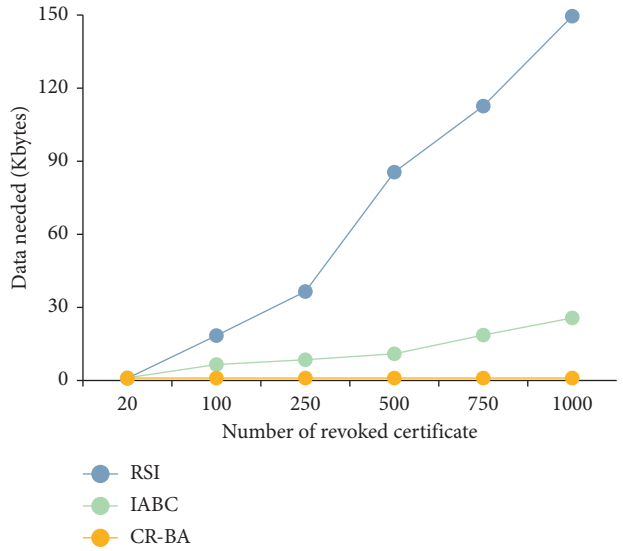


Figure 12: Amount of data needed to exchange to provide a response on the revocation status of a revoked certificate.

```
Input: accValue, member, witness, key
Output: f_result or false//validation response
(1) ChaincodeStub stub = ctx.getStub ();
(2) byte[] ojectBytes = stub.getState (Accumulator.class.getSimpleName ());
(3) Accumulator accValue = deserialize (ojectBytes); //deserialize the accumulator object
(4) for member in MEMBER//member are in MEMBER
(5) Boolean verify = acc.verifyMembership (accValue, member, witness, acc.getN ()); //Verify that the current certificate exists
(6) result ← Acc.verify (member); //get the verification result
(7) if (result = = 0)
(8)                 Acc ← Acc.Delete(member); //delete the certificate
(9)             witness = acc.proveMembership (sha256 (cert, n)); //compute new accumulator value and new revocation factor
(10)                f_result ← provemembership (sha256 (cert, n)); //f_result is the result
(11)                Return f_result; //return update result
(12)        else
(13) Return false;
```

Algorithm 2: Update algorithm, MemWitUp.

```
Input: accValue, member, witness, key
Output: true or false//validation response
(1) ChaincodeStub stub = ctx.getStub ();
(2) byte [] ojectBytes = stub.getState(Accumulator.class.getSimpleName ());
(3) Accumulator accValue = deserialize (ojectBytes); //deserialize the accumulator object
(4) Acc ← FindBlockchainContract; //get the accumulator value
(5) result ← Acc.verify(member);//verify member is in the accumulator
(6) if (result = = 0) {
(7)         return true; //verify successfully
(8) else
(9) return false;
```

Algorithm 3: Verification algorithm, verify.

Table 1: Comparison of advantages and disadvantages of different methods.

| Methods | Certificate management | False-positive rate | Certificate change |
|---|---|---|---|
| Reference [8] IABC | Exist | Exist | Exist |
| Reference [9] RSI | Not exist | Exist | Not exist |
| Reference CAB | Exist | Not exist | Exist |

possible insider threats caused by compromised nodes in the distribution system.

*5.2. Lower Revocation Storage Costs.* This solution reduces the cost of storing certificates after introducing the accumulator to the certificate storage. CA acts as an accumulator administrator, aggregating certificates into accumulator values. Accumulator represents the entire set of elements with a single value, and the accumulator value and witness are the sizes of the RSA modulus. Accumulator allows the witness to prove whether the element is in the set, independent of the number of elements in the element.

*5.3. No False Positives.* The verification algorithm will always return 1 for all honestly generated keys, all honestly calculated cumulative values, and evidence. It is difficult to find membership evidence for elements that do not belong to the set, and it is also challenging to find evidence of nonmembership, which is collision-free. Accumulator has undeniability, indicating that computing two conflicting pieces of evidence for elements $x \in X$ or $x \notin X$ is computationally infeasible.

## 6. Experiment

*6.1. Experimental Environment.* The experimental model is deployed on a PC with the following configuration: Intel(R) Core(TM) i7-10750H CPU, 16 GB RAM. Ubuntu 18.04 OS, Hyperledger Fabric v1.4.2, chain code using Golang 1.14.12, Docker version number 19.03.2. The chain code uses Golang 1.14.12 and Docker version number 19.03.2.

*6.2. Results and Discussion.* Before the experimental test, 5000 digital certificates are created in batches. To avoid the contingency of experimental results, repeat five times to calculate the average value. Moreover, it compares certificate revocation methods using bloom and cuckoo filters. Table 1 indicates the comparison of the advantages and disadvantages of different methods. Figure 9 compares the average query time of the revoked certificate. Figure 10 represents the cost comparison of using different blockchain certificate storage methods. Figure 11 represents the amount of data required to respond to an unrevoked certificate's revocation status. Figure 12 represents the amount of data required to respond to a revoked certificate's revocation status.

Figure 9 shows the results of the time required when querying the status of a revoked certificate. Query time of revocation certificate for CR-BA is 35.01 ms on average, and it is also about 1 second to query 10,000 certificates. RSI takes more time than IABC and CR-BA. Because additional verification is required when querying the status of a revoked certificate. As the number of certificates increases, so does the additional validation required. IABC has the most efficient query but suffers from false positives (providing a positive response while the certificate is still not revoked). Our approach has no false positives, and the response time is within reasonable limits. Figure 10 shows the comparison of certificate storage costs. RSI stores a complete Bloom filter on the blockchain, and IABC stores a cuckoo filter on the blockchain, both as a complete array. CR-BA stores value, which is a small summary as mentioned before. As seen from the figure, the storage consumption of certificate storage in our approach is about half of the other methods compared to others.

Figure 11 shows the amount of data needed to exchange in response to the revocation status of a nonrevoked certificate. All three approaches relied on a simple request and verified response that does not change much when the number of certificates increases. Figure 12 shows the amount of data needed to exchange to respond to a revoked certificate's revocation status. RSI achieves the worst performance. Since RSI needs to download the RSI structure, after the filter provides a positive response, it must ensure no false-positive response. It needs to download all LRSI structures. There are as many LRSI as revoked certificates, so it will take more time and data volume. IABC requires a similar amount of data as our method, which does not change much as the certificate increases. However, in this validation scenario, IABC has the problem of false positives.

It can be seen from the above performance tests that the certificate query time of [9] increases linearly with the increase of test set size. The time consumption of this article fluctuates very little with the increase in the number of certificates, but it lags behind the query speed of reference [8]. References [8, 9] suffer from the probability of misjudgment, but this article does not have this problem. Moreover, compared with other methods, certificate storage in this article consumes less storage. With the increase of blockchain data, the cost of blockchain certificate data storage can be reduced, and it has certain validity and feasibility.

## 7. Conclusion

This article first analyzes the current certificate revocation mechanism's shortcomings and expounds relevant knowledge of blockchain and accumulators. A public key infrastructure certificate revocation scheme based on blockchain and accumulator is proposed to address problems existing in the current certificate status query method. It take advantages of the efficient and verifiable features of the accumulators and features that support dynamic addition and removal of member elements. It builds a certificate containing the revocation factor by generating a revocation

accumulator in the smart contract. The certificate's fingerprint is written into the accumulator as a member value, which improves query efficiency when the data on the chain is huge and reduces certificate storage overhead.

## Data Availability

The data used to support the findings of this study are included within the article.

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## Acknowledgments

This work was supported in part by the National Natural Science Foundation of China, under Grant no. 61373162, and Sichuan Provincial Science and Technology Department Project, under Grant no. 2022YFG0161.

## References

[1] J. P. Monteuuis, B. Hammi, and E. Salles, "Securing pki requests for c-its systems," in *Proceedings of the 2017 26th International Conference on Computer Communication and Networks (ICCCN)*, July 2017.

[2] R. Wang, J. He, and C. Liu, "A privacy-aware PKI system based on permissioned blockchains," in *Proceedings of the 2018 IEEE 9th international conference on software engineering and service science (ICSESS)*, November 2018.

[3] J. Leng, M. Zhou, and J. L. Zhao, "Blockchain security: a survey of techniques and research directions," *IEEE Transactions on Services Computing*, vol. 1, p. 1, 2020.

[4] J. Leng, G. Ruan, P. Jiang et al., "Blockchain-empowered sustainable manufacturing and product lifecycle management in industry 4.0: a survey," *Renewable and Sustainable Energy Reviews*, vol. 132, Article ID 110112, 2020.

[5] J. Leng, S. Ye, M. Zhou et al., "Blockchain-Secured smart manufacturing in industry 4.0: a survey," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 1, pp. 237–252, 2021.

[6] S. Jinshan and L. Ru, "Survey of blockchain access control in internet of things [J]," *Journal of Software*, vol. 30, no. 06, pp. 1632–1648, 2019.

[7] L. Tan, H. Xiao, K. Yu, M. Aloqaily, and Y Jararweh, "A blockchain-empowered crowdsourcing system for 5G-enabled smart cities," *Computer Standards & Interfaces*, vol. 76, Article ID 103517, 2021.

[8] L. Tan, H. Xiao, and X. Shang, "A blockchain-based trusted service mechanism for crowdsourcing system," in *Proceedings of the 2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, May 2020.

[9] L. Tan, N. Shi, C. Yang, and K. Yu, "A blockchain-based access control framework for cyber-physical-social system big data," *IEEE Access*, vol. 8, Article ID 77215, 2020.

[10] J. Leng, D. Yan, Q. Liu et al., "ManuChain: combining permissioned blockchain with a holistic optimization model as Bi-level intelligence for smart manufacturing," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 1, pp. 182–192, 2020.

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

[12] 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.

[13] K. Rabieh, M. M. Mahmoud, K. Akkaya, and S Tonyali, "Scalable certificate revocation schemes for smart grid AMI networks using bloom filters," *IEEE Transactions on Dependable and Secure Computing*, vol. 14, no. 4, pp. 420–432, 2017.

[14] S. Medury, A. Skjellum, and R. R. Brooks, "Scraaps: X. 509 certificate revocation using the blockchain-based scrybe secure provenance system," in *Proceedings of the 2018 13th International Conference on Malicious and Unwanted Software (MALWARE)*, IEEE, Nantucket, MA, USA, October 2018.

[15] S. Huang, L Jian, and F. A. N. Bingbing, "IABC: a cross-domain authentication method based on blockchain and cuckoo filter," *Journal of Chinese Computer Systems*, vol. 41, no. 12, p. 6, 2020.

[16] J. Benaloh and M. D. Mare, "One-way accumulators: a decentralized alternative to digital signatures," in *Proceedings of the Workshop on the Theory and Application of of Cryptographic Techniques*, Springer, New York, NY, USA, May 1993.

[17] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proceedings of the Workshop on the theory and application of cryptographic techniques*, Springer, New York, NY, USA, June 1984.

[18] R. Perlman, "An overview of PKI trust models," *IEEE network*, vol. 13, no. 6, pp. 38–43, 1999.

[19] S. Tuecke, V. Welch, and D. Engert, *Internet X. 509 Public Key Infrastructure (PKI) Proxy Certificate Profile*, Proposed Standard, 2004, https://www.ietf.org/rfc/rfc3820.txt.

[20] R. Hunt, "PKI and digital certification infrastructure," in *Proceedings of the Proceedings Ninth IEEE International Conference on Networks, ICON 2001*, IEEE, Bangkok, Thailand, October 2001.

[21] L. Harn and J. Ren, "Generalized digital certificate for user authentication and key establishment for secure communications," *IEEE Transactions on Wireless Communications*, vol. 10, no. 7, pp. 2372–2379, 2011.

[22] D. Cooper, S. Santesson, and S. Farrell, *Internet X. 509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, 2008, https://rfc-editor.org/rfc/rfc5280.txt.

[23] R. Housley, W. Polk, and W. Ford, *Internet X. 509 public key infrastructure certificate and certificate revocation list (CRL) profile*, NIST: National Institute of Standards and Technology: Gaithersburg, MD, USA, 2002.

[24] D. A. Cooper, "A more efficient use of delta-CRLs," in *Proceedings of the 2000 IEEE Symposium on Security and Privacy. S&P 2000*, pp. 190–202, IEEE, Berkeley, CA, USA, May 2000.

[25] S. Boeyen, T. Howes, P. Richard, and X. Internet, "509 public key infrastructure LDAPv2 schema," *IETF Request For Comments*, vol. 2587, pp. 99–104, 1999.

[26] J. I.-W. U. Jing and L. I. N. Jing-Qiang, *FENG DENG-GUO Technologies on Public Key Infrastructure*, Science Press, Beijing China, 2008.

[27] Y. U. A. N. Xue, *Research on Certificate Revocation Mechanisms*, Institute of Software Chinese Academy of Sciences, Beijing China, 2004.

[28] S. Micali, *Efficient Certificate Revocation*, Technical Memo MIT/LCS/TM-542b, Massachusetts Institute of Technology, Laboratory for Computer Science, MA, USA, 1996.

[29] M. Naor and K. Nissim, "Certificate revocation and certificate update," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 4, pp. 561–570, 2000.

[30] P. C. Kocher, *On Certificate Revocation and Validation*-Springer, Heidelberg, Germany, 1998.

[31] J. Leng, P. Jiang, K. Xu et al., "Makerchain: a blockchain with chemical signature for self-organizing process in social manufacturing," *Journal of Cleaner Production*, vol. 234, pp. 767–778, 2019.

[32] Q. Hu, M. R. Asghar, and N. Brownlee, "Certificate Revocation Guard (CRG): an efficient mechanism for checking certificate revocation," in *Proceedings of the IEEE 41st Conference on Local Computer Networks (LCN). IEEE*, Dubai, UAE, November 2016.

[33] T. Hewa, A. Braeken, M. Ylianttila, and L. Madhusanka, "Multi-access edge computing and blockchain-based secure telehealth system connected with 5G and IoT," in *Proceedings of the IEEE Global Communications Conference (Globecom)*, Taipei, Taiwan, December 2020.

[34] Z. Lu, Q. Wang, and G. Qu, "BARS: a blockchain-based anonymous reputation system for trust management in VANETs," in *Proceedings of the 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, August 2018.

[35] N. Malik, P. Nanda, and A. Arora, "Blockchain based secured identity authentication and expeditious revocation framework for vehicular networks," in *Proceedings of the 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, August 2018.

[36] X. Feng, Q. Shi, Q. Xie, and L. Liu, "An efficient privacy-preserving authentication model based on blockchain for VANETs," *Journal of Systems Architecture*, vol. 117, Article ID 102158, 117 pages, 2021.

[37] Q. Wang, R. Li, and Q. Wang, "Poster: transparent certificate revocation for CBE based on blockchain," in *Proceedings of the Poster Session of 41st IEEE Symposium on Security and Privacy (SP)*, IEEE, Guangdong, China, June 2020.

[38] C. Lin, D. He, X. Huang, N. Kumar, and K. KR. Choo, "BCPPA: a blockchain-based conditional privacy-preserving authentication protocol for vehicular ad hoc networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 12, pp. 7408–7420, 2021.

[39] S. Yao, J. Chen, K. He, R. Du, T. Zhu, and X. Chen, "PBCert: privacy-preserving blockchain-based certificate status validation toward mass storage management," *IEEE Access*, vol. 7, pp. 6117–6128, 2019.