IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 7, 2018, accepted December 7, 2018, date of publication December 27, 2018,
date of current version January 16, 2019.

Digital Object Identifier 10.1109/ACCESS.2018.2889898

PBCert: Privacy-Preserving Blockchain-Based Certificate Status Validation Toward Mass Storage Management

SHIXIONG YAO[1], JING CHEN[1,2], KUN HE[1], RUIYING DU[1,3], TIANQING ZHU[4], AND XIN CHEN[1]

[1]School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China
[2]Shenzhen Institute of Wuhan University, Shenzhen 518057, China
[3]Collaborative Innovation Center of Geospatial Technology, Wuhan 430079, China
[4]School of Software, University of Technology Sydney, Ultimo, NSW 2007, Australia

Corresponding author: Jing Chen (chenjing@whu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant U1836202, Grant 61772383, Grant 61572380, and Grant 61702379, and in part by UnitedData Holding group Co. Ltd.

ABSTRACT In the recent years, the vulnerabilities of conventional public key infrastructure are exposed by the real-world attacks, such as the certificate authority's single-point-of-failure or clients' private information leakage. Aimed at the first issue, one type of approach is that multiple entities are introduced to assist the certificate operations, including registration, update, and revocation. However, it is inefficient in computation. Another type is to make the certificate information publicly visible by bringing in the log servers. Nevertheless, the data synchronization among log servers may lead to network latency. Based on the second approach, the blockchain-based public key infrastructure schemes are proposed. Through these type of schemes, all the certificate operations are stored in the blockchain for public audit. However, the issue of revoked certificates' status storage is worth paying attention, especially in the setting with massive certificates. In addition, the target web server that a client wants to access is exposed in the process of certificate status validation. In this paper, we propose a privacy-preserving blockchain-based certificate status validation scheme called PBCert to solve these two issues. First, we separate the revoked certificates control and storage plane. Only the minimal control information (namely, certificate hashes and related operation block height) is stored in the blockchain and it uses external data stores for the detailed information about all revoked certificates. Second, we design an obscure response to the clients' certificate status query for the purpose of privacy preserving. Through the security analysis and experiment evaluation, our scheme is significant in practice.

INDEX TERMS Public key infrastructure, blockchain, revocation mechanism, privacy-preserving.

I. INTRODUCTION

With the development of information technology in recent years, a lot of tasks have been fulfilled over network. More and more people are becoming increasingly aware of the security of their information transporting in network. So an increasing number of websites serve over Secret Socket Layer (SSL) or Transport Layer Security (TLS). They move away from the plaintext HTTP protocol to securer HTTPS. TLS is to guarantee the data confidentiality and integrity for the vast majority of on-line connections. Public Key Infrastructure (PKI) is used to authenticate the identity of web server by public key digital certificate. The certificates issuing, update and revocation are authorized by a third trusted party called Certificate Authority (CA).

However, real-world attacks have indicated the vulnerability of the existing CA [1]. Intuitively, if a CAs has been attacked, a malicious web server may get a verifiable certificate. Due to a corruption of the trusted root, tens of thousands of clients may suffer losses in finance or personal privacy. Aimed at this vulnerability, researches have done plenty of research work. Through investigation, we classify these proposals into two different types. The main idea of one group is to *disperse the trust of CAs* and the other is to make the CAs' *behavior public visible*. In the first group, multiple CAs or other entities are brought in to assist certificate registration, update, and revocation for the purpose of diminishing the trust of CAs. For instance, in ARPKI, Basin *et al.* [2] introduce multiple CAs to sign and validate certificates in a serial mode.

In TriPKI, Chen *et al.* [1] utilize threshold signature among CAs and DNSs to form a tripartite PKI system, in which, any party with misbehavior will be detected by other parties. In the second group, notaries, integrity log servers (ILS) or validators are brought in to make the certificates public visible and detect CAs' misbehavior. For example, Google proposes Certificate Transparency [3] which brings in ILSs to record certificates for public audit. Continually, AKI [4], CIRT [5], and Sovereign Key (SK) [6] and so on enhance the security of PKI system based on notaries. Compared with the first group, the second is more practical. Because the introduced entities participating in certificate authorization increase the computation complexity and any one of participants' failure may incur the failure of certificates authorization. However, certificate operation behavior receiving public audit makes the misbehavior detectable quickly.

Inspired by the characteristics of blockchain, such as decentralization, public audit and tamper-proof, some schemes based on blockchain are presented to make behavior related to certificate operation public visible, for instance, IKP [7], CertChain [8], Certcoin [9] and so on. We focus on CertChain which is a typical scheme fallen into the second group. The others will be described in section II. In CertChain, Chen *et al.* [8] provide a public audit PKI system with certificate operation history traceability and revocation verification. What's more, the certificates' operations are recorded in blockchain which provides public and self audit. However, there are still three issues unsolved. **1)** ***Confined block size.*** The scale of certificates is up to 10 million or even more, the size of a block is not enough to hold all the revoked certificates even utilizing bloom filters. **2)** ***Excessive data redundancy.*** Responsible for recording the latest certificates' status, valid and revoked certificates bloom filters update with new block generation. However, the block updating interval is about 6.7s. The number of certificates whose status changed in this slot is rare. In other words, these two types of bloom filters are the same or hardly unchanged in continuous multiple blocks, which gives rise to excessive date redundancy. **3)** ***Browsing privacy leak.*** As same as the conventional certificate status lookup by the way of OCSP (Online Certificate Status Protocol), in CertChain, the objective web servers accessed by Clients may be leaked when the clients request for certificates revocation verification.

In this paper, aimed at these issues mentioned above, we present a privacy-preserving blockchain-based certificate status validation toward massive storage management, which is called PBCert. In detail, we separate the storage and control plane. In other words, all the revoked certificates are stored in OCSP servers and the control information about these revoked certificates are recorded in blockchain. By this way, the block size cannot be restricted by the number of revoked certificates and the data redundancy is negligible. Aimed at the privacy-preserving in certificate status validation, we design an obscure response to clients' certificate revocation queries. In summary, we make the following contributions:

- To the best of our knowledge, this is the first work to support large scale certificates management based on blockchain with low storage cost.
- We separate the revocation certificates control and storage to solve the issue of block size limitation and the excessive revocation information redundancy in blockchain.
- We design an obscure response method to preserve the clients' privacy when they validate whether the objective certificates have been revoked from a OCSP server.
- We analyze the security of PBCert in theory. We also implement a proof-of-concept prototype and evaluate the performance of PBCert in practice.

The remainder of this paper is organized as follows. Section II reviews the literature related to the PKI. In section III, we give a description about the background of PBCert and the preliminaries. We present the details of our PBCert in section IV, and analyze the security of PBCert in section V. Section VI evaluates the performance of this scheme. Finally, we conclude this paper in section VII.

II. RELATED WORK

As the fundamental of secure network communication, PKI has been paid more attention. Aimed at the following issues, such as the single-point-of-failure, inefficient revocation mechanisms, and the privacy-preserving in certificate status validation, researchers have done a lot of work. Based on these research results, we give a mind map about the overview of secure PKI as shown in FIGURE 1. The details are given below.

A. DISPERSE THE TRUST OF CA

Before establishing a secure TLS connection with a web server, the client must verify the Certificate Authority (CA) signature on this server's certificate. CA as the third trust party is responsible for signing and issuing certificates. All the CAs form a trust chain from a root CA to multiple intermediate CAs to commercial CAs. No matter which CA is compromised, all the certificates issued by it after it been controlled cannot be trusted. Therefore, the intuitive method to solve this issue is to disperse the trust of CA. Basin *et al.* [2] put forward ARPKI to decrease the trust of one CA by introducing several CAs transfer the trust. In detail, these CAs receive, check, sign and send the certificates to the next CA in line. Nevertheless, this trust transference model is inefficient. If one CA is compromised, the process of certificate registration may need to restart. Chen *et al.* [1] design a tripartite PKI system called TriPKI by bringing in the DNSs and Integrity Log servers (ILSs) to assist CA in managing certificates. Through the threshold signature algorithm among multiple entities, the system can tolerate the failure of a certain number of CAs or other entities. However, the implementation cost of this scheme is expensive in practice. Syta *et al.* [10] propose the CoSi which utilizes an existing cryptographic multi-signature method to support thousands of witnesses via signature aggregation. When a client verifies the aggregated

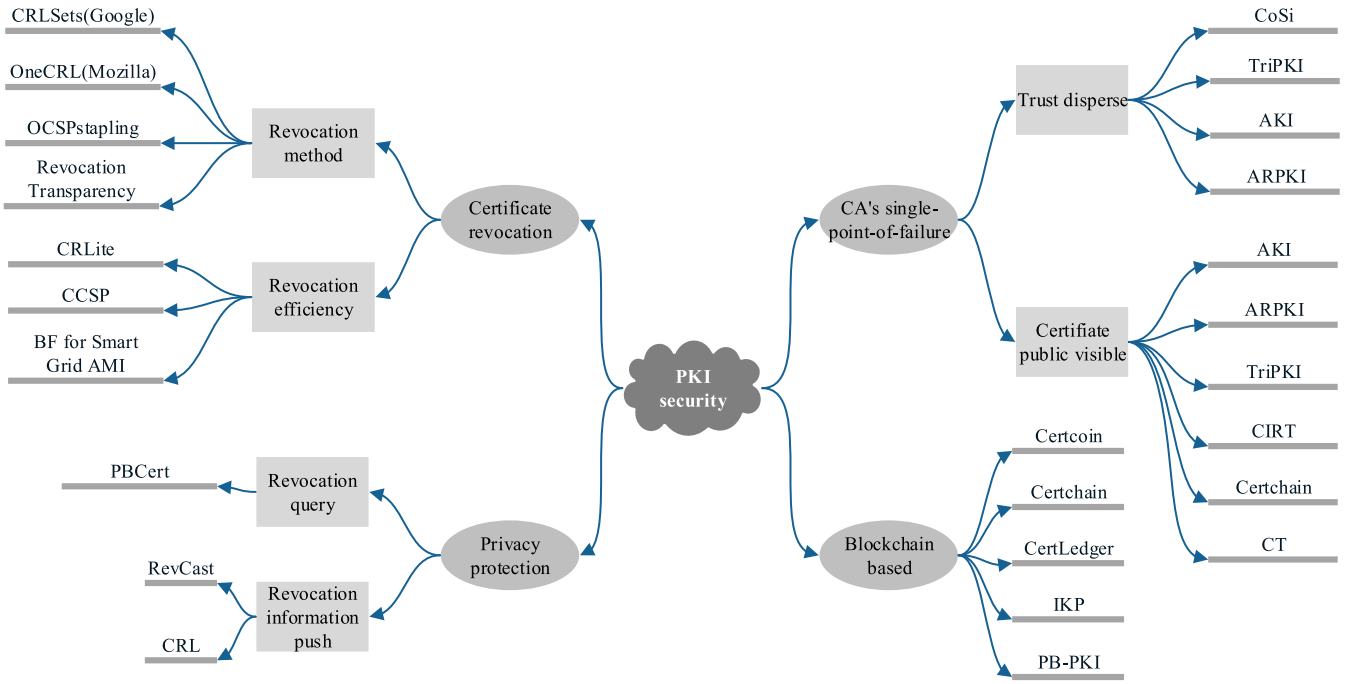


FIGURE 1. Overview of secure PKI.

signature on the web server's certificate, it needs to get a collective public keys of the witnesses who participate in signing this certificate. Therefore, ensuring the authenticity of all witness' identities and public keys will be the bottleneck of each client.

B. CERTIFICATE PUBLIC VISIBLE

In conventional PKI, certificates are not public visible except for the corresponding domains and CAs, so a malicious web server's certificate signed by a compromised CA is difficult to be detected. Therefore, in 2013, Google firstly propose the Certificate transparency (CT) [3] which induces CT logs. The logs record certificates in the form of merkle hash tree which supports efficient proof of existing. Based on CT, a lot of schemes are presented to reinforce the security of PKI system. Ryan [5] put forward the Certificate Issuance and Revocation Transparency (CIRT) to enhance CT with an efficient revocation mechanism. Chen *et al.* [1], Basin *et al.* [2], Kim *et al.* [4], and Ryan [5] are inspired by CT and bring in the integrity log servers to record certificates and revocation information. Yu *et al.* [11] propose a Distributed Transparent Key Infrastructure (DTKI) which reduces the oligopoly of service providers and allows verification of the behavior of trusted entities.

C. CERTIFICATES REVOCATION ISSUES

OCSP stapling advocates that the certificate and its status are stapling when a client and a web server establish TLS connection. The clients neither passively download the CRL nor actively request the OCSP server for certificates' statuses. The web servers are responsible for providing signed information for indicating the statuses of certificates. Chen *et al.* [12] present a network-coding-based method which can be used to issue revocation certificates in wireless mesh network. Schulman *et al.* [13] present a broadcast system called RevCast, which disseminates revoked certificates in a timely and private manner. The CAs or revocation servers push the revoked certificates to all clients actively by FM radio. Indeed, this method preserves clients' browsing privacy, but all clients must install FM radio receiver terminals. Szalachowski *et al.* [14] propose PKISN to solve the too-big-to-be-revoked problem triggered by an attacked or compromised CA, which may lead to massive collateral damage. PKISN makes CAs to revoke their own certificates after a certain point in time and to remain previously-signed valid certificates. Larisch *et al.* [15] design CRLite to aggregate revocation information and store them in a space-efficient filter cascade structure without false positive rate. Rabieh *et al.* [16] also employ bloom filters for certificate revocation in smart grid advanced metering infrastructure (AMI) communications. Chariton *et al.* [17] present a CCSP to supplement OCSP and OCSP Stapling. In this scheme, the revoked certificates are packed in a bitmap. Each bit of the bitmap indicates the revocation status of a single certificate. In order to decrease the storage and transmission costs, the bitmap is compressed in space and time dimensions.

D. BLOCKCHAIN BASED CERTIFICATE MANAGEMENT

Inspired by the characteristics of blockchain, such as the decentralization, public audit and tamper-proofing, blockchain has been introduced in PKI system to solve some issues. Fromknecht *et al.* [9] present the Certcoin to build

TABLE 1. Comparison of different public key infrastructures based on security, availability and deployability.

		CT	AKI	ARPKI	TriPKE	IKP	CRLite	CertChain	PBCert
Security	MitM attack resistance	N	Y	Y	Y	N	N	Y	Y
	Interval of CA compromised	months	LUP	LUP	0	LUP	BCP	BCP	BCP
	Temple-proof of certificates records	N	N	N	N	Y	N	Y	Y
	Rate of parties that must be compromised for successful attack	1/1	2/3	n/n	t/n	$(\frac{1}{2}n)/n$	1/1	$(\frac{1}{2}n)/n$	$(\frac{1}{2}n)/n$
	Client browsing privacy-preserving	N	N	Y	N	N	N	N	Y
Availability	Certificate operation duration of unavailability	0	LUP	seconds	0	0	-	BCP	BCP
	Log server compromised duration of unavailability	days	LUP	LUP	LUP	-	-	0	0
	Interval of CA compromise detection	-	LUP	LUP	0	BCP	-	BCP	BCP
	Impersonate attack detection of domain	Y	Y	Y	Y	N	N	Y	Y
	Misbehavior reaction automatically	N	N	N	N	Y	N	N	N
	Incentive mechanism among entities	N	N	N	N	Y	N	Y	Y
	Certificate history traceability	N	N	N	N	N	N	Y	Y
	Certificate revocation efficiency	L	L	L	L	L	H	M	M
	Concurrency of certificate operation requests	M	M	M	L	L	M	M	H
Deployability	CA changed required	N	N	Y	Y	N	Y	Y	Y
	Domain changed required	N	Y	Y	Y	N	Y	Y	Y
	client-side plug-in required	N	Y	Y	Y	N	N	Y	Y
	Other entities required	Log	Log	Log	Log, DNS	node	-	BK	BK, OCSP
	Data synchronization among entities	N	Y	Y	Y	Y	N	Y	Y

LEGEND FOR TABLE VALUES:
MitM attack: Man in the middle attack, LUP: Log server's Update Period, BCP: Block Confirmation Period
n: a system parameter, L: low, M: middle, H: high, BK: bookkeeper, Log: log server

a PKI that ensures identity retention. In other words, Certcoin prevents a web server from registering a public key based on others identity registered already. The core idea of this scheme is to maintain a public ledger of domains and their associated public keys. Ali *et al.* [18] propose the Blockstack which enables clients to register human-readable names and associate public keys along with additional data to these names. It does not need any central or trusted party. Chen *et al.* [8] put forward the CertChain to make the certificate operations public and self audit. CertChain records the history of each certificate operation from registration to revocation. It also makes all revoked certificates public visible. Matsumoto and Reischuk [7] present Instant Karma PKI (IKP) which stimulates CAs to correctly issue certificates and detectors to quickly report unauthorized certificates. They design domain certificate policy and reaction policy in smart contact for the purpose of defining and processing misbehavior automatically. Kubilay *et al.* [19] design the CertLedger which is an architecture with certificate transparency and revocation transparency based on blockchain. Axon and Goldsmith [20] consider a privacy-aware blockchain-based PKI called PB-PKI. It provides unlinkable key updates and user-controlled disclosure. Al-Bassam presents an alternative PKI system based on a web-of-trust model and a smart contract on the Ethereum blockchain. It is used to detect the rogue certificates when they are published. Szalachowski [22] puts forward PADVA (Persistent and Accountable Domain Validation) which combines the advantages of previous works and leverages a blockchain to provide new features.

E. COMPARISON

In this part, we give a comparison about different PKI schemes based on security, availability, and deployability as shown in TABLE 1. For security entry, we pay attention to the classical attack, the cost of a successful attack, data security, and the privacy-preserving. Next, we give a comparison about the availability entry. In this entry, what we firstly consider is how long the period for a certificate operation cannot work after a CA or Log server is compromised. Then, the misbehavior detection, reaction and incentive mechanisms are taken into consideration. The certificate history traceability and revocation efficiency are worthy of attention. At last, we focus on the deployability of these different schemes. The main concern is that whether the entities need to be changed based on the traditional PKI system.

III. BACKGROUND AND DESIRED PROPERTIES

A. CERTCHAIN DESIGN

We review the CertChain [8] in more detail for two reasons. First, our scheme is inspired by CertChain's design and employs some of its concepts. Second, our scheme addresses several shortcomings in CertChain. In CertChain, Chen et al.

design a certificate management based on blockchain, which supports public audit and efficient revocation checking. In this scheme, the interactive information flows of certificate operations involve following three entities:

1 **Certificate authorities (CAs)** are responsible for issuing X.509 certificates to authenticate domains and generating certificate operations which are stored in blockchain by bookkeepers.
2 **Bookkeepers** maintain the blockchain and record the certificate operations and revocation information in blockchain.
3 **Domains** are web servers request certificates from CA for TLS connection with clients.

A certificate operation (registration, update, and revocation) is launched from a domain to a CA (certificate revocation operation can be initiated by a CA) through the operation request. The CA signs the certificate contained in the request, and generates an operation then sends it to a bookkeeper. The bookkeeper stores it into the blockchain and returns the block height to the CA. The signed certificate and block height are sent to the domain.

In order to provide **public audit**, the authors design a data structure called `CertOper` to express the certificate operation. The format of `CertOper` is similar to the format of the standard X.509 certificate. It contains some new fields such as *operation type*, *timestamp & notafter*, *last operation height* and so on. All certificate operations corresponding to each domain make up an operation chain for public audit, which reflects the integrated operation history.

For the purpose of **efficient revocation validation** in blockchain, the authors design the **DCBF** (dual counting bloom filters) to record all the revocation information in the latest block. Via DCBF, a client requests an arbitrary bookkeeper to check whether the target certificate is revoked or not.

CertChain weakness. Storing certificate operations benefits the public and self audit indeed, however, to the best of our knowledge, the revocation information stored in blockchain is not the best solution. The forward traceability is the inherent characteristic of blockchain, but certificate revocation operation always follows the registration or update. It is easy to get the previous operation according to the current status, however, it's hard to get latest status of a certificate. In CertChain, the authors adopt a method that records all certificates statuses in the latest block. By this way, anyone at any time can get the revocation status from the latest block. However, the size of one block is limited while there is no upper limit of the number of certificates. When the amount of revoked certificates reaches up to ten million or even more, the size of DCBF will exceed the capacity of one block. What's more, since the interval of time between two blocks is too short, the DCBFs in continuous blocks are the same or negligible changed. In other words, the storage space utilization of the whole blockchain is too low.

What's more, in CertChain, clients' browsing privacy in revocation query has not been considered. As well as in conventional PKI, during the process of certificate status query, the client needs to send the objective certificate to a bookkeeper which may be an honest but curious entity.

B. ADVERSARY MODEL

In our scheme, the goals of an adversary attacking PKI system are: 1) compromising a CA without being detected; 2) issuing unauthorized certificate for malicious domain; 3) deceiving clients with an inauthentic certificate revocation status. We assume that an adversary is able to forge, modify, eavesdrop message. It can compromise any entity but not all. The OCSP servers are honest but curious.

Moreover, we make some standard cryptographic assumptions, such as, the adversary cannot forge signatures without the target server's private key. It cannot control more than 51% bookkeepers.

C. DESIRED PROPERTIES

Here are the properties that we expect of a satisfactory certificate management system:

- **large-scale:** The scale of certificates does not affect the storage property of blockchain.
- **Privacy:** Clients can obtain the accurate certificate status information without sacrificing their privacy.
- **Authenticity:** Only legitimate parties can create a revocation message for a certificate, but this message can be verified by everyone.
- **Transparency:** Revocation information must be publicly accessible and persistent. If a revoked certificate is successfully issued, it is impossible for an interested party to claim that the certificate is still valid.

IV. PBCert DESIGN

A. OVERVIEW

PBCert retains most good characteristics in CertChain, such as public and self audit, certificate history traceability and the fairness consensus. At the same time, it makes up for the deficiencies of block size limitation and privacy-preserving in certificate status validation. Based on CertChain, we bring in the OCSP servers for certificate revocation. We improve the function of OCSP server in conventional PKI and make it compatible with CertChain. In details, we separate the revocation information's storage and control plane and design an efficient obscure response to revocation query for the purpose of preserving clients' privacy.

The framework of PBCert is shown in FIGURE 2. There are five types of entities in our scheme including CAs, bookkeepers, domains, OCSPs, and clients. A CA is responsible for certificate issuing, update, revoking, and operation generating. Bookkeepers maintain the blockchain and store certificate operations. OCSP servers take charge of storing the revoked certificates and responding certificate status query from clients. Domains and clients establish security connections through domains' certificates issued by CAs. Clients can validate the certificate operation and status from an

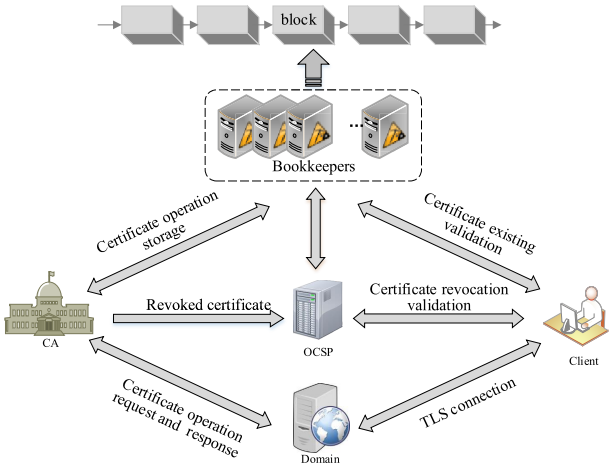


FIGURE 2. Framework of PBCert.

arbitrary bookkeeper. The detailed certificate operations will be given in the following sections.

B. CERTIFICATE REGISTRATION AND UPDATE

1) REGISTRATION AND UPDATE FLOW

Generally, the certificate registration is often launched from a domain to a CA. In practical, the CA needs to check a lot of information related to the domain's identity off-line and this step is outside of considering in this paper. We assume the domain is a legitimate web server. The flow of certificate registration and update is shown in FIGURE 3. The CA receives the domain's registration request, it signs the certificate and generates a certificate registration operation called `CertOper` (details about `CertOper` will be described in IV-B2). Then, the CA sends `CertOper` to an arbitrary bookkeeper. The bookkeeper broadcasts this message to all others. When a bookkeeper is selected as a leader, it records all buffered certificate operations into the new block and broadcasts the new blockchain. If these records have been confirmed, each bookkeeper returns the block height of the corresponding certificate operation to the CA. The CA sends a response to the domain's request with a signed certificate and the corresponding height value.

The process of certificate update operation is similar with registration. In general, the CA does not need to check the domain's identity information, and sometimes, the CA only expands the expiration data. So, we do not give unnecessary details about the process of update operation.

2) DATA PROCESS

In PBCert, the certificate operation follows the designing in CertChain. It is a new data structure defined to express three certificate operation including *registration*, *update*, and *revocation*. The format of it is similar with the standard X.509 certificate. Besides the traditional fields such as `Version No.`, `Signature Algorithm ID`, `Signature Value`, `Extension Field`, it also contains some customized fields including `Subject Name`, `Operation Type`, `Time & NotAfter`, `Current certificate hash`, `Last Operation Height`. Especially, through the `Last operation Height`, the certificate operation history of each domain can be traceable. This characteristic is in favor of public and self audit.

C. CERTIFICATE REVOCATION

1) REVOCATION FLOW

When a domain's private key is leaked, the domain needs to apply for revoking the previous certificate and request a new one. The certificate revocation flow is shown in FIGURE 4. A domain applies for certificate revocation from a CA. The CA sends the revoked certificate to OCSP server. In the meantime, it generates the `CertOper` with the type of revocation, signs and sends it to a bookkeeper. As the same with registration and update operation, this `CertOper` will be recorded into the blockchain, and the block height will be sent to the OCSP server rather than the CA. Then, the OCSP server generates `CertRevo` (details about `CertRevo` will be described in IV-C2) and add this information into the merkle hash tree. The root of this tree will be written in the latest block.

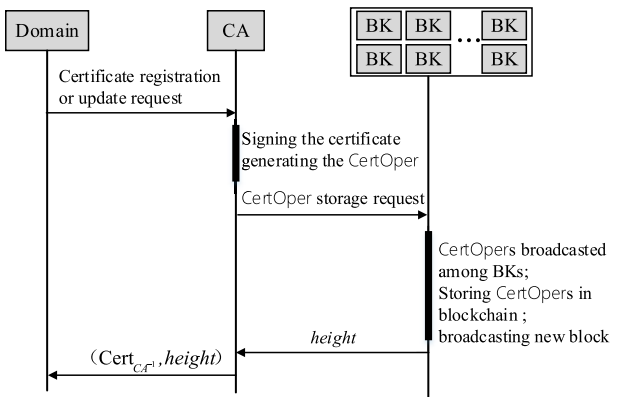


FIGURE 3. Certificate registration and update flow. In this figure, CA denotes certificate authority, BK denotes the bookkeeper. All bookkeepers maintain the blockchain.

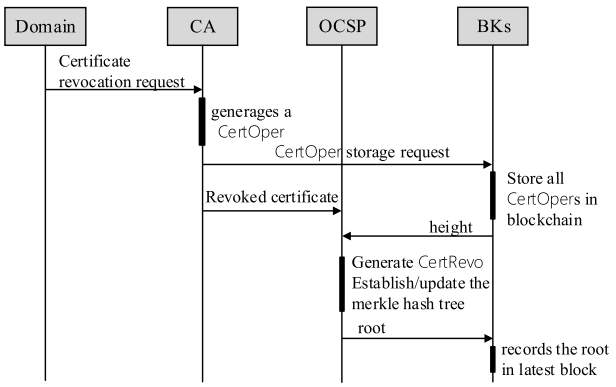


FIGURE 4. Certificate revocation flow. It is worth to note that certificate revocation can be launched by a CA directly. In this setting, the first information flow can be ignored.

TABLE 2. Structure of a `CertRevo`.

Field
Version Number
Subject Name
Revocation Operation Height
Timestamp
Certficiate hash
Signature Algorithm ID
Signature Value
Extension Field

2) SEPARATION OF CONTROL AND STORAGE PLANE

PBCert separates the revoked certificate control and storage planes. The control plane consists of certificate operations and all OCSP servers' root hashes related to the information of all revoked certificates. The storage plane is responsible for storing the detail information of all revoked certificates issued by CAs. We define a new data structure called `CertRevo` to describe the information of each revoked certificate. It is designed in the format exhibited in TABLE 2 and all the fields are explained as follows. Each `CertRevo` is signed by the public keys of the OCSP server.

- `Version Number`, `Signature Algorithm ID`, `Signature Value`, `Extension Field` are the same as X.509 certificate.
- `Subject Name`: the name of a domain whose certificate has been revoked;
- `Revocation Operator height`: the block height of the subject's certificate revocation operation;
- `Timestamp`: the time when this certificate has been revoked;
- `Certificate Hash`: the hash of a subject domain's certificate which is used for the process of certificate revocation validation;

As shown in FIGURE 5, in the control plane, the blockchain works for two purposes. The one is to store the sequence of certificate operations. The other is to record root hashes from all OCSP servers. Root hashes are treated as special certificate operations stored in the latest block.

In the storage plane, all revoked certificates are stored in OCSP servers, and they are signed by the key of the respective owner of a CA. By storing revoked certificates outside of the blockchain, our scheme can provide revocation query for large scale of certificates. Clients do not need to trust the storage plane because they can verify the integrity of the revoked certificates in the control plane. In order to insure the tamper-proofing of `CertRevos`, a merkle hash tree is constructed and the root of this tree is recorded in the latest block.

D. CERTIFICATE VALIDATION

1) VALIDATION FLOW

When a client wants to establish a secure TLS connection with a domain, it will receive the domain's certificate in the handshake protocol. Then it needs to check whether the certificate is valid from four aspects: ① signature verification, ② expiration date checking, ③existence checking (the certificate operation must be stored in blockchain), ④ revocation query.

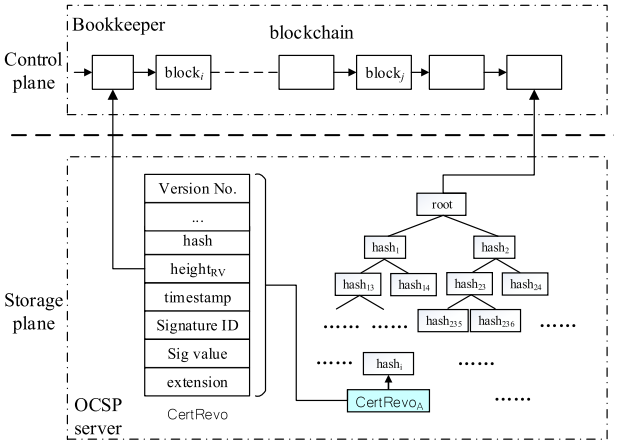


FIGURE 5. Separation of control and storage plane. The control plane is maintained by bookkeepers, and the storage plane is maintained by every OCSP server.

The information flow of certificate validation is shown in FIGURE 6. The former two steps are accomplished locally by a client via a conventional way, the latter two steps have recourse to an OCSP server and a bookkeeper. Therefore, we introduce the latter two steps in details. According to the operation height value, the clients query an arbitrary bookkeeper for the existing of corresponding certificate operation. For the last step, the client sends a request of the revocation *ReqRev* to an OCSP server. The former looks up the database and gets some revoked certificates satisfied client's requirement. Through an obscurely processing, the OCSP server gives a response to the client. The processing detail is described in IV-D2. If all four steps validation are correct, the client can communicate with the domain securely. Note that the last two steps can be executed in parallel.

2) PRIVACY-PRESERVING IN REVOCATION QUERY

In order to preserving clients' privacy in revocation query, we design an obscure response to a client's request. From

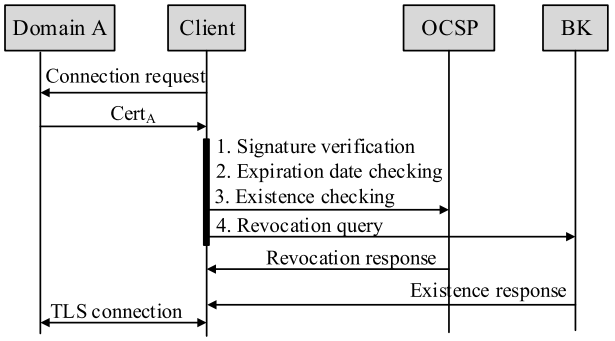


FIGURE 6. Certificate validation flow. Aimed at the client certificate existence checking, it looks up the corresponding certificate operation according to the block height attached in client's checking information. Then, it gives the certificate existence response to the client.

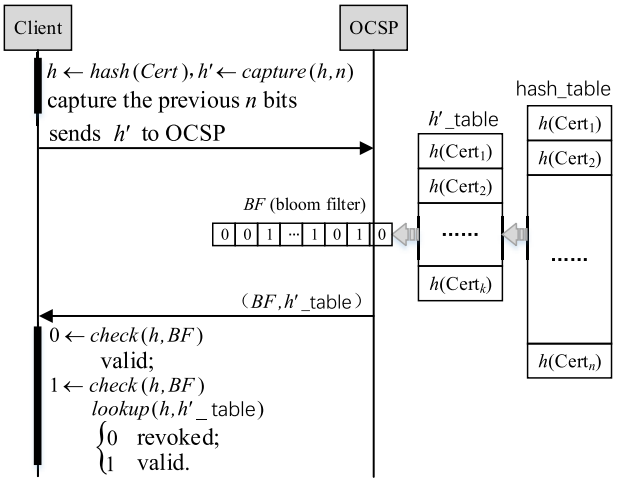


FIGURE 7. Certificate revocation obscure request and response flow.

the validation flow in IV-D1, the client sends the information about the desired certificate to OCSP server. In this step, the client needs to get the certificate hash h and captures a certain length of the hash value that is h'. As shown in FIGURE 7, after receiving h', the OCSP server consults the *hash_table* to get the all the certificates whose hashes value match h'. Then it adds these certificates into a table called h'*_table* and inserts all of them into a bloom filter called BF. The revocation information (BF, h'*_table*) is sent to the client. The client firstly checks whether the hash h of the objective certificate is in the bloom filter. If it is not in, then the objective certificate in valid. Otherwise, the client needs to look up h'*_table*. If h is in, the objective certificate is revoked, otherwise it is valid. Note that, because of the false positive rate of bloom filter, if h is in the BF, checking the h'*_table* is necessary. In addition, if the OCSP server finds none of certificates matching h', it responds to the clients that the objective certificate is valid directly. It is worth to note that the bloom filter is used to increase the efficiency of revocation status request and response.

V. SECURITY ANALYSIS

Theorem 1: The adversary cannot attack a CA to issue a certificate which could pass the certificate validation for a malicious web server without being detected.

Proof: The goal of an adversary is to issue a certificate for a malicious web server by controlling a CA. This type of certificate is called as the unauthorized certificate. In PBCert, a certificate that can pass the validation process must satisfy four conditions displayed in certificate validation flow in IV-D1. Because of the third step, all the certificate operations must be stored in blockchain to receive public audit. The malicious web server's unauthorized certificate issued by a compromised CA is also visible in public. The manager of a CA can consult the whole blockchain to detect whether there is any unauthorized certificate issued by itself but the identity of the subject has not been checked. If there are some authorized certificate operations recorded in blockchain, the involved CA may be attacked. Then the CA needs to determine the time of the attack. More precisely, this time is when an unauthorized certificate is observed in the blockchain. All the certificates issued after attacking need to be checked.

Therefore, setting a rational frequency of self audit, the attack against CA will be detected quickly. ■

Theorem 2: The adversary cannot personate a legal web server to get a certificate that could pass the certificate validation without being detected.

Proof: In the conventional PKI system, an adversary can personate a legal web server by registering a public key under this web server's already-registered identity. This attack is difficult to detect. Because all the certificates are not public visible. In PBCert, any domain can scan the blockchain to detect the impersonation attack via self audit. ■

Theorem 3: The adversary cannot forge the certificate status to deceive clients without being detected.

Proof: From the certificate validation flow in IV-D1, a certificate's status is valid or revoked is determined by a OCSP server's response. An adversary may disturb the TLS connection between a client and web server. For instance, the adversary can prevent a web server to provide legal service by issuing a forge revocation status to clients. It also can change a revoked certificate's status to make a malicious web server whose certificate has been revoked still provide service.

In PBCert, each revoked certificate is stored by the data structure called `CertRevo` which records the revocation operation in blockchain. From FIGURE 5, we can see that these revocation items are stored in a merkle hash tree, and the root of this tree is recorded in blockchain. Since each `CertRevo` contains the block height of the revocation operation stored in blockchain, the accurate status of a certificate can be traced by the corresponding operation. And all the revocation information cannot be tampered. In some setting, in order to improve clients' website experience, the browser vendor makes clients browse the webpage before revocation validation. When the certificate is detected to be revoked by background, a warning pop-up will be given by the browser. In PBCert, the clients can establish the TLS connection when the certificate passes the revocation validation. And the accurate status confirmation of the certificate is detected in background. If the certificate is revoked, the TLS connection must be terminated. ■

Theorem 4: Compared with conventional OCSP and Certchain, PBCert can preserve clients' privacy without losing the certificate status validation efficiency.

Proof: In conventional OCSP revocation mechanism, clients get the accurate certificate status with sacrificing the privacy of accessing web server, as well as in CertChain. In PBCert, the queried certificate has been preprocessed by a one-way hash function and captured n bits. After receiving a certificate status request, the OCSP server cannot recover the integrity certificate or bits. Through a hash matching

from the whole table of revoked certificates, the OCSP can get a subtable and generate a bloom filter to insert these matching certificates. A reasonable value of n can guarantee the confusion of a subtable. Even if the number of elements in subtable is small, the OCSP cannot confirm the objective certificate. For instance, if the required certificate is valid, it cannot appear in the matched revoked certificates list. Note that, the computation complexity of the objective certificate lookup in PBCert is $O(n)$ as well as in conventional OCSP. Compared with CertChain, the efficiency of certificate status validation depends on the third step (operation existence checking). The bloom filter is used to improve the efficiency of status confirmation for clients. Therefore, PRCert preserve clients' privacy with efficient status validation. ■

VI. PERFORMANCE EVALUATION

A. IMPLEMENTATION

We have developed a prototype implementation of PBCert. We implement the CA with OpenSSL. The main process is written in Node.js. Each CA provides services including certificate registration, update, revocation, lookup and validation by calling the uniform REST API interface. The bookkeeper is established by Ethereum. The services related to the blockchain operations call the main interfaces such as:(1) `sendTransaction`, (2)`filter.watch`, (3)`getTransactin`, (4)`getBlock` and so on. We implement the domain by the express framework. The main process is written in Node.js. The domain's network load balancing is achieved by nginx reverse proxy. A client is built by Firefox Developer Edition. We install the designed plug-in for the certificate validation service with an OCSP server and a bookkeeper. Based on the existing Online Certificate State Protocol (OCSP), we design a new data structure called `CertRevo` which is defined and coded by ASN.1(Abstract Syntax Notation One). In this prototype, the bloom filter and the merkle hash tree are realized by calling the nmp source. The certificate operation calls the X.509 library. All the entities connect with each other by a secure TLS connection.

For simplicity, entities of the same party are simulated in one PC. Hence, a CA is implemented by four virtual CPUs, 16G RAM, and Ubuntu LTS 16.04 64bit operation system, as well as the domain and OCSP server. A bookkeeper is implemented with 2 virtual CPUs, 8GB RAM and Ubuntu LTS 16.04.

B. A PERFORMANCE COMPARISON OF REVOCATION MECHANISM BETWEEN PBCert AND CONVENTIONAL PKI

In the revocation process, PBCert is a trade-off between CRL and OCSP. So in this part, we will make a comparison with these two schemes.

In terms of storage cost, the average size of a CRL in conventional PKI is about 51KB [23]. In PBCert, the size of the h'_table is smaller than the CRL intuitively. Because the h'_table is a small subset of all the revoked certificates. For example, a CRL with the size of 45KB contains about 840 revoked certificates. When we set the value of n as 8, we can measure that the size of the h'_table is about 112 Byte and the size of bloom filter is 9 Byte. For another, if the CRL is 1.2MB, it contains about 499675 revoked certificates. When the value of n is 16, the sizes of h'_table and bloom filter are 176 Byte and 14 Byte respectively.

Since the size of h'_table is adjustable, from TABLE 3, we can find that the size is smaller, the objective certificate is more obvious, and the risk of privacy leakage is higher, vice versa. Therefore, getting a rational value of n is essential.

TABLE 3. The number of revoked certificates in h'_table with different n under different total number of revoked certificates.

T_{RCert}	Value of n	N_{RCert} in h'_table
499675	12	136
	14	25
	16	11
	18	1
	20	1
840	8	7
	10	3
	12	2
	14	1
	16	1

LEGEND FOR TABLE VALUES:
T_{RCert}: Total number of revoked certificates
N_{RCert}: The number of revoked certificates in h'_table
n: The number of captured bits.

In terms of the computation cost, we measure the time for an OCSP server to get the certificates matching h' and to build a bloom filter for these selected revoked certificates. We assume that no matter how large the number of revoked certificates in an OCSP server, through the rational value of n, the h'_table consists of several hundred certificates. In this process, we can find that creating of h'_table produces few latencies. Because the subtable is built during traversing the whole hash table composed of all revoked certificates. A bloom filter is built by the hashes of these revoked certificates in h'_table and this process costs about 10ms. The average latency of revocation validation in conventional OCSP is about 250ms [23]. Therefore, the delay of our scheme is the nearly the same, so this result is acceptable.

C. THE PERFORMANCE INDEXES IN THE PROCESS OF CERTIFICATE VALIDATION

For a PKI system, the efficiency of certificate validation process is the most concerned issue. Because it affects the user experience in the website browsing. Therefore, we will measure the performance of certificate validation from three aspects.

Firstly, the storage and communication costs of certificate validation are determined by the size of the (h'_table, BF). In our implementation, the average size of (h'_table, BF) is about 4KB with a h'_table containing several hundred of items. Compared with the CRL in conventional PKI, our result is more efficient in storage cost.

Secondly, due to the OSCP server's obscure response, the client needs to determine the accurate status of the objective certificate. The main computation cost is reflected in the lookup of the bloom filter and h'_table. Considering the signature verification delay and the expiration date checking latency, the computation cost of a client is about 9 ms. For a client, this process delay is insignificant.

Thirdly, from validation flow IV-C1, step 3 and step 4 in certificate validation process involve communication from a client to a bookkeeper and an OCSP server. These two steps can be carried out in parallel. That is to say, the network latency is determined by the step with longer delay. In step 3, according to a given certificate hash and height value, a bookkeeper looking up a certificate operation will cost about 443 ms. And the result feeding back costs about 3ms. Therefore, the existing validation from a bookkeeper costs about 446 ms. In step 4, the OCSP needs to generate a h'_table and a bloom filter according to a given h', and a client must query the bloom filter even the h'_table to get the accurate status of a certificate. From the above results, OCSP needs about 135 ms to generate a subtable and a bloom filter. The client needs about 147 ms to get the status from (BF, h'_table). Therefore, considering the network delay, the latency is about 283 ms.

In conclusion, the certificate validation delay is about 459 ms. This result is comparable to the CertChain. Because the most time-consumption steps are certificate existence checking. In other words, we improve the performance in storage and privacy-preserving without more time-consumption.

D. COMPARED WITH CERTCHAIN

In this part, we focus on the size of one block and the blockchain with fixed number of blocks. We make a comparison between CertChain and PBCert under the setting that the scale of certificates is about one million and the revoked certificates is in the range of 1% to 10%.

From FIGURE 8, we can find that the block size of PBCert is stable under an incremental rate of revoked certificates, but it grows notably in CertChain. The reason is that the revoked certificates information is not recorded in blockchain directly. Due to the separation of storage and control plane, the revoked certificates are stored in OCSP server by the merkle hash tree with the `CertRevo` as the leaf node. Only the hash root treated as a special certificate operation stored in the latest block. The number of root hashes is related to the number of OCSP server which is not changed frequently. But in CertChain, all revoked certificates are stored in the latest block by two counting bloom filters. Even if the bloom filter is an efficient storage structure, if the certificate scale is as large as ten million or even more, the size of one block is not enough to hold one counting bloom filter. Therefore, the PBCert is more rational in the setting with large scale certificates.

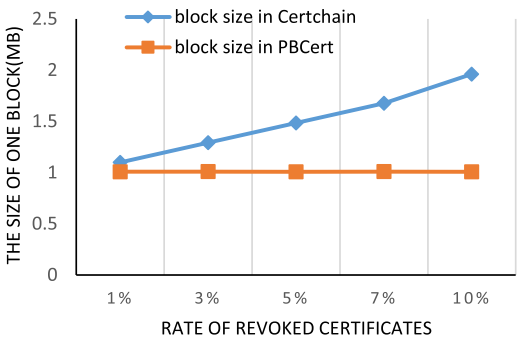


FIGURE 8. Under the setting with about one million certificates with different rate of revoked certificates, the block size comparison between PBCert and CertChain.

The FIGURE 9 reflects the size of blockchain in a certain number of blocks. We simulate a setting that the scale of certificates is about one million and the rate of revoked certificate is about 5%. The rate of certificate operations in unit interval is uniform. In other word, the speed of block generation is enough to contain the operations. In this setting, we can find that the line of CertChain is increase more rapidly than the line of PBCert. Therefore, the storage of blockchain in PBCert is more efficient. For another, the data redundancy degree is lighter. That is to say, even if an OCSP server does not receive new revoked certificate in a block interval, the new block only needs to restore the OCSP's root value. By contrast, CertChain must rebuild the whole dual counting bloom filter if only one certificate status changes.

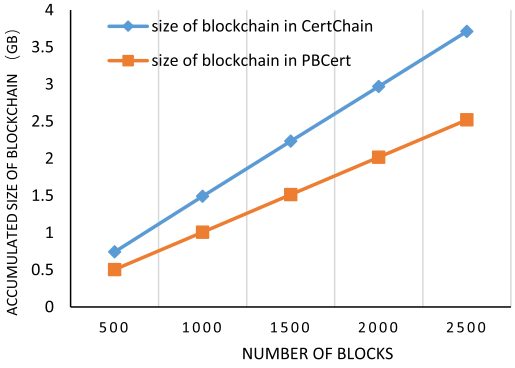


FIGURE 9. Under the setting with about one million certificates and the rate of revoked certificates is 5%, the accumulated size of blockchain in incremental number of blocks.

In conclusion, we solve the issue of large scaled revoked certificates and excessive data redundancy in blockchain. At the same time, we preserve the clients' browsing privacy. And from the results, the improvement about CertChain is meaningful.

VII. CONCLUSION

To solve the revoked certificate storage scale issue and preserve clients' privacy, we present a scalable certificate revocation privacy-preserving PKI system in this paper. In details, firstly, we separate the revoked certificates control

and storage plane by bringing in the conventional OCSP server. The OCSP servers store the revoked certificate in the format of `CertRevo` which is a new data structure related to certificate revocation operation. They also maintain a merkle hash tree related to all `CertOper` and store the root into the blockchain. Secondly, we design an obscure response to the clients' revocation query for the purpose of privacy-preserving. Through the security analysis and the experiment evaluation, PBCert is significant in real world.

REFERENCES

[1] J. Chen, S. Yao, Q. Yuan, R. Du, and G. Xue, "Checks and balances: A tripartite public key infrastructure for secure Web-based connections," in *Proc. IEEE IEEE Int. Conf. Comput. Commun. (INFOCOM)*, May 2017, pp. 1–9.

[2] D. Basin, C. Cremers, T. H.-J. Kim, A. Perrig, R. Sasse, and P. Szalachowski, "ARPKI: Attack resilient public-key infrastructure," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2014, pp. 382–393.

[3] B. Laurie, A. Langley, and E. Kasper, *Certificate Transparency*, document RFC 6962, 2013. [Online]. Available: https://tools.ietf.org/html/rfc6962

[4] T. H.-J. Kim, L.-S. Huang, A. Perring, C. Jackson, and V. Gligor, "Accountable key infrastructure (AKI): A proposal for a public-key validation infrastructure," in *Proc. WWW*, 2013, pp. 679–690

[5] M. D. Ryan, "Enhanced certificate transparency and end-to-end encrypted mail," in *Proc. NDSS*, 2014, pp. 1–14. [Online]. Available: http://dblp.dagstuhl.de/rec/bib/conf/ndss/Ryan14

[6] S. Josh. *Sovereign Keys: An EFF Proposal*. Accessed: Dec. 18, 2014. [Online]. Available: http://www.m9development.com/sovereign-keys-eff-proposal/

[7] S. Matsumoto and R. M. Reischuk, "IKP: Turning a PKI around with decentralized automated incentives," in *Proc. IEEE S&P*, May 2017, pp. 410–426.

[8] 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 *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2018, pp. 2060–2068.

[9] C. Fromknecht, D. Velicanu, and S. Yakoubov, "CertCoin: A namecoin based decentralized authentication system 6.857 class project," Massachusetts Inst. Technol., Cambridge, MA USA, Tech. Rep., 2014. [Online]. Available: http://courses.csail.mit.edu

[10] E. Syta *et al.*, "Keeping authorities 'honest or bust' with decentralized witness cosigning," in *Proc. IEEE S&P*, May 2016, pp. 526–545.

[11] J. Yu, V. Cheval, and M. Ryan, "DTKI: A new formalized PKI with verifiable trusted parties," *Comput. J.*, vol. 59, no. 11, pp. 1695–1713, Jul. 2016.

[12] J. Chen, K. He, R. Du, M. Zheng, Y. Xiang, and Q. Yuan, "Dominating set and network coding-based routing in wireless mesh networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 2, pp. 423–433, Feb. 2015.

[13] A. Schulman, D. Levin, and N. Spring, "RevCast: Fast, private certificate revocation over FM radio," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2014, pp. 799–810.

[14] P. Szalachowski, L. Chuat, and A. Perrig, "PKI safety net (PKISN): Addressing the too-big-to-be-revoked problem of the TLS ecosystem," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P)*, Mar. 2016, pp. 407–422.

[15] J. Larisch, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, and C. Wilson, "Crlite: A scalable system for pushing all TLS revocations to all browsers," in *Proc. IEEE S&P*, May 2017, pp. 539–556.

[16] K. Rabieh, M. M. A. Mahmoud, K. Akkaya, and S. Tonyali, "Scalable certificate revocation schemes for smart grid AMI networks using Bloom filters," *IEEE Trans. Dependable Secure Comput.*, vol. 14, no. 4, pp. 420–432, Jul. 2017.

[17] A. A. Chariton, E. Degkleri, P. Papadopoulos, P. Ilia, and E. P. Markatos, "CCSP: A compressed certificate status protocol," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, May 2017, pp. 1–9.

[18] M. Ali, J. Nelson, R. Shea, and M. J. Freedman, "Blockstack: A global naming and storage system secured by blockchains," in *Proc. USENIX Annu. Tech. Conf. (USENIX ATC)*, 2016, pp. 181–194. [Online]. Available: https://www.usenix.org/conference/atc16/technical-sessions/presentation/ali

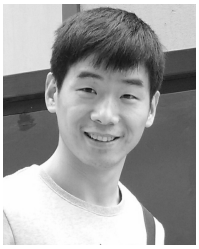
[19] M. Y. Kubilay, M. S. Kiraz, and H. A. Mantar. (Jun. 2018). "CertLedger: A new PKI model with certificate transparency based on blockchain." [Online]. Available: http://arxiv.org/abs/1806.03914

[20] L. Axon and M. Goldsmith, "PB-PKI: A privacy-aware blockchain-based PKI," in *Proc. 14th Int. Joint Conf. e-Bus. Telecommun. (SCITEPRESS)*, 2017, pp. 1–8.

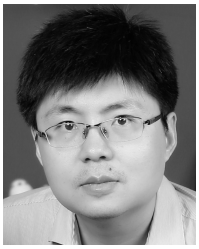
[21] M. Al-Bassam, "SCPKI: A smart contract-based PKI and identity system," in *Proc. ACM Workshop Blockchain, Cryptocurrencies Contracts (BCC)*, 2017, pp. 35–40.

[22] P. Szalachowski. (Apr. 2018). "Blockchain-based TLS notary service." [Online]. Available: http://arxiv.org/abs/1804.00875

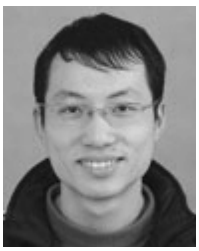
[23] Y. Liu *et al.*, "An end-to-end measurement of certificate revocation in the Web's PKI," in *Proc. ACM IMC*, 2015, pp. 183–196.



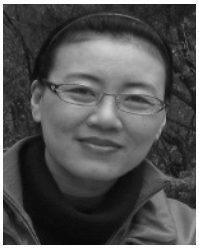
SHIXIONG YAO received the M.S. degree in computer science from Wuhan University, Wuhan, China, in 2014, where he is currently pursuing the Ph.D. degree. His research interests include network coding, public key infrastructure, and blockchain.



JING CHEN is currently a Professor with the Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, Wuhan University. He has published more than 90 research papers in many international journals and conferences, including the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, the IEEE TRANSACTIONS ON MOBILE COMPUTING, the IEEE TRANSACTIONS ON COMPUTERS, INFOCOM, SECON, and TrustCom. His research interests include the areas of network security and cloud security.



KUN HE received the Ph.D. degree in computer science from Wuhan University. He has published research papers in the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, the *International Journal of Communication Systems*, *Security and Communication Networks*, and the IEEE TrustCom. His research interests include cryptography, network security, mobile computing, and cloud computing.



RUIYING DU received the B.S., M.S., and Ph.D. degrees in computer science from Wuhan University, Wuhan, China, in 1987, 1994, and 2008, respectively. She is currently a Professor with the School of Cyber Science and Engineering, Wuhan University. Her research interests include network security, wireless network, and mobile computing.

TIANQING ZHU received the B.Eng. and M.Eng. degrees from Wuhan University, China, in 2000 and 2004, respectively, and the Ph.D. degree in computer science from Deakin University, Australia, in 2014. She was a Lecturer with the School of Information Technology, Deakin University, from 2014 to 2018. She is currently a Senior Lecturer with the School of Software, University of Technology Sydney, Australia. Her research interests include privacy preserving, data mining, and network security.

XIN CHEN received the bachelor's degree in information security from Wuhan University, Wuhan, China, in 2017, where he is currently pursuing the master's degree. His research interests include blockchain and computer network.