


Blockchain-Based Certificate Transparency and Revocation Transparency

Ze Wang, Jingqiang Lin , Senior Member, IEEE, Quanwei Cai, Qiong Xiao Wang, Daren Zha, and Jiwu Jing, Member, IEEE

Abstract—Traditional X.509 public key infrastructures (PKIs) depend on trusted certification authorities (CAs) to sign certificates, used in SSL/TLS to authenticate web servers and establish secure channels. However, recent security incidents indicate that CAs may (be compromised to) sign fraudulent certificates. In this article, we propose blockchain-based certificate transparency (CT) and revocation transparency (RT) to balance the absolute authority of CAs. Our scheme is compatible with X.509 PKIs but significantly reinforces the security guarantees of a certificate. The CA-signed certificates and their revocation status information of an SSL/TLS web server are published by the subject (i.e., the web server) as a transaction in the global certificate blockchain. The certificate blockchain acts as append-only public logs to monitor CAs' certificate signing and revocation operations, and an SSL/TLS web server is granted with the cooperative control on its certificates. A browser compares the certificate received in SSL/TLS negotiations with the ones in the public certificate blockchain, and accepts it only if it is published and not revoked. We implement the prototype system with Firefox and Nginx, and the experimental results show that it introduces reasonable overheads.

Index Terms—Blockchain, certificate transparency, certificate revocation, public key infrastructure, trust management

1 INTRODUCTION

IN X.509 public key infrastructures (PKIs), a certification authority (CA) signs certificates to bind the public key of a server to its identity (e.g., a DNS name). These certificates are used in SSL/TLS [20], [30] to authenticate web servers. Trusting the CAs, browsers obtain the servers' public keys from CA-signed certificates in SSL/TLS negotiations, to establish secure channels.

However, security incidents indicate that CAs are not so trustworthy as they are assumed to be. CAs may sign fraudulent certificates due to misoperations [64], [95], reckless identity validations [63], [81], [93], intrusions [15], [34], [90], or government compulsions [25], [78]. Typical fraudulent certificates bind a DNS name (e.g., www.facebook.com or www.gmail.com) to key pairs held by counterfeit web servers [4], [15], [25]. Then, the counterfeit servers will successfully launch man-in-the-middle (MitM) attacks, even if a

browser follows the strictest steps of certificate validation [16] to establish SSL/TLS sessions.

Certificate transparency (CT) was proposed to enhance the accountability of CA operations, by *append-only* public logs [52]. A CA-signed certificate is publicly recorded in the log servers; otherwise, a browser rejects it in SSL/TLS negotiations. So a fraudulent certificate will be observed by interested parties, especially the owner of the DNS name (or the web server). Independent auditors periodically verify the integrity of the records to ensure that they are append-only, i.e., a (fraudulent) certificate will never be deleted or modified after appended.

CT follows a reactive philosophy. It depends on interested parties to observe fraudulent certificates after they have been recorded in log servers. So a fraudulent certificate may be accepted by browsers before it is observed. Moreover, the detection of deleted or modified records in the logs relies on the verification by extra auditors.

In this paper, we propose a blockchain-based storage to construct public logs for certificates, which is inherently append-only. A blockchain is a peer-to-peer (P2P) storage chain of blocks, each of which includes transactions [67]. Each P2P node always considers the longest chain that it ever received as the valid version, and subsequent blocks are appended to the currently-valid chain and broadcast to other nodes. To append a block, a nonce needs to be found by brute force, which is called *mining*. The computation cost of block mining prevents attackers from forging another longer chain (i.e., modifying the certificate logs), after the valid version has been accepted by a majority of nodes.

In the proposed blockchain-based scheme, a CA-signed certificate is published by its subject (i.e., the corresponding SSL/TLS web server) in a *certificate transaction* in the *global*

- Z. Wang, Q. Cai, and D. Zha are with the State Key Laboratory of Information Security (LOIS), Institute of Information Engineering, and the Data Assurance and Communication Security Research Center (DCS Center), Chinese Academy of Sciences, Beijing 100864, China. E-mail: {wangze, caiquanwei, zhadaren}@iie.ac.cn.
- J. Lin and Q. Wang are with the State Key Laboratory of Information Security (LOIS), Institute of Information Engineering, and the Data Assurance and Communication Security Research Center (DCS Center), Chinese Academy of Sciences, Beijing 100864, China, and also with the School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China. E-mail: {linjingqiang, wangqiong Xiao}@iie.ac.cn.
- J. Jing is with the School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 100049, China. E-mail: jwjing@ucas.ac.cn.

Manuscript received 30 July 2018; revised 31 Mar. 2019; accepted 8 Mar. 2020. Date of publication 30 Mar. 2020; date of current version 17 Jan. 2022.

(Corresponding author: Jingqiang Lin.)

Digital Object Identifier no. 10.1109/TDSC.2020.2983022

certificate blockchain. In the Bitcoin blockchain [67], a transaction is signed by some Bitcoin owner(s). In the proposed scheme, each web server has a *publishing key pair*, different from the ones bound in certificates, to publish its certificates (i.e., sign the certificate transactions).

This design of certificate publication allows an SSL/TLS web server to control its certificates cooperatively with CAs. A certificate is signed by CAs but published by the subject. It shares the same spirit with PoliCert [84] and trust assertions for certificate keys (TACK) [59] that a server is involved in the validation of its certificates, explicitly or implicitly. We implement this control as the subject-signed transactions in the public blockchain-based certificate logs.

In our scheme, SSL/TLS web servers compose an interdependent community, to actively tame the absolute authority of CAs in traditional X.509 PKIs. First, the publishing key of a web server is initially certified by a certain number of the interdependent web servers and also a CA. So CAs are unable to publish a certificate in the blockchain, without the consent of the community of web servers. Second, the certified publishing key of each web server is also recorded in the blockchain. It means that the publication of a certificate is also publicly accountable. That is, the certificate blockchain is maintained publicly as append-only logs to monitor both CAs' and web servers' operations. Compared with other security-enhanced certificate services in practice [2], [26], [27], [39], [52], this design gives the web servers a chance to cooperatively control their certificates, but not depend on other single entity to achieve this goal.

Each certificate transaction has a validity period, shorter than those of the published certificates; and a web server periodically publishes its certificates. It brings the following benefits. First, CAs' revocation operations are recorded and revocation transparency (RT) [51], [73] is achieved. A certificate will not be published any more after it is revoked. When an unexpired but revoked certificate is absent from the next transaction, the corresponding CA-signed certificate revocation list (CRL) file or online certificate status protocol (OCSP) response is included in the transaction instead.

More importantly, the validity period of certificate transactions introduces the incentives to mine the blocks. At any time, some certificate transactions will expire soon. So the web servers that need to publish certificates in new transactions, are willing to cooperatively mine the block including these transactions, or pay other servers for the mining. We design the special crypto-currency (called *cert-coin* in this paper) that is generated, consumed, transferred, and destroyed in the certificate blockchain, to facilitate the mining. For the sake of simplicity, in the following, we assume a web server also works as a miner to cooperatively mine blocks, but a miner can be a node specialized in mining blocks in the community.

Browsers utilize the certificate blockchain to validate the certificates in SSL/TLS negotiations. A server certificate is accepted only if it is published in an unexpired transaction, so a fraudulent certificate signed by compromised CAs but not published in the blockchain will be rejected. A browser incrementally downloads the certificate blockchain from the P2P storage network of web servers. This download may be performed when there is no SSL/TLS negotiation.

This scheme protects browsers against the MitM and impersonation attacks using fraudulent certificates. It seamlessly integrates CT [52], RT [51], [73] and subject-controlled certificate publication [59], [84] into the blockchain-based certificate logs. This scheme reinforces the security guarantees of an X.509 certificate, with the following features.

- *Blockchain-based CT and RT.* CAs' certificate signing and revocation operations are recorded in the public logs. The operation results (i.e., certificates, CRL files and OCSP responses) are stored publicly in the inherently append-only blockchain.
- *Transparent subject-controlled certificate publication.* An SSL/TLS certificate is signed by CAs but published by its subject. Web servers are involved in certificate services, to balance the absolute authority of CAs in traditional PKIs. Moreover, the publication is also publicly recorded.
- *Preservation of client privacy.* Browsers download (the block headers of) the certificate blockchain, to validate certificates. The download is uniform and indistinguishable for all browsers, not leaking any privacy information about its SSL/TLS sessions, such as the visited web servers and the occurrence time.
- *Compatibility with traditional X.509 PKIs.* The CAs operate the same as the ones in traditional X.509 PKIs, to sign certificates and revocation status information [16]. Standard X.509 certificates, CRL files and OCSP responses are published in the certificate blockchain.
- *Incremental deployment.* It does not require all web servers or browsers in the Internet to work together. Even when the community consists of only a number of web servers, the security of their certificate validation is enhanced in the SSL/TLS negotiations with the browsers that support the proposed scheme. A web server or browser that supports our scheme performs interoperably with another one that does not, and vice versa.

We implemented the prototype system with Nginx and Firefox. The Nginx server sends its certificate transactions to browsers as SSL/TLS extensions, and the browser validates the received certificates with the help of the certificate blockchain. The analysis based on the real-world statistics and the experimental results show that our scheme introduces reasonable overheads, in terms of storage, certificate validation delay, communication, transaction verification cost, and incentive cost.

The remainder of this paper is organized as follows. Section 2 introduces the security model and design goal of our scheme, and the system details are described in Section 3. Then, the proposed scheme is analyzed in Section 4, and the prototype system is evaluated in Section 5. Section 6 includes related work, and Section 7 concludes this paper.

2 THREAT MODEL AND DESIGN GOAL

Attackers attempt to impersonate an SSL/TLS web server using fraudulent certificates, and a successful attack means that browsers accept the fraudulent certificates in SSL/TLS negotiations. The attackers could compromise some CAs that

are trusted by browsers, to sign fraudulent certificates binding the target web server's DNS name to any key pair.

We assume that the attackers could directly compromise a number of web servers except the target server, while the majority of servers keep honest (i.e., follow their specifications of our scheme). We do not assume that the attackers could directly compromise the target web server. In such cases, the attackers impersonate the server directly using the compromised key pair, and no fraudulent certificate is needed. Such attacks cannot be countered by certificate management, and it is out of the scope of this paper.

For a compromised server, the attackers could use all its long-term secrets (e.g., key pairs) to generate any messages. In particular, the attackers might collude with malicious servers to certify a fraudulent publishing key. That is, we consider the attack scenario where the publishing key pair is held by attackers and fraudulent certificates are signed by compromised CAs at the same time.

The attackers do not hold computation resources to arbitrarily manipulate the blockchain [29], and all cryptographic primitives are secure. The attackers cannot block the network for a long time to take attack actions; that is, honest entities communicate with each other in a loosely synchronized manner.

We aim to protect browsers against the MitM and impersonation attacks using fraudulent certificates. In the attack case that CAs are compromised to sign fraudulent certificates, if a browser follows our scheme to validate server certificates in SSL/TLS negotiations, the authenticated peer is ensured to be the legitimate server of the visited DNS name. Even in the extreme case that a fraudulent certificate has been published in the blockchain by a compromised publishing key pair, the browser rejects this fraudulent certificate and it will be recovered by countermeasure transactions in the certificate blockchain.

3 BLOCKCHAIN-BASED CT AND RT

3.1 Overview

There are two types of certificate transactions. A *Type-I transaction* is signed by a web server using its publishing key pair, to periodically publish certificates. When a certificate expires and is updated, the new one will be included in the next transaction. If a certificate is revoked, it will be excluded from the next transaction, and the corresponding CRL file or OSCP response will be included instead. *Type-II transactions* are used to initialize or reset publishing key pairs. When a DNS name (or web server) is initially introduced into this community, its publishing key is signed by a number of web servers (called *certifiers*) using their publishing key pairs. The publishing key may be reset by another Type-II transaction, if it is compromised or lost.

Certificate transactions labeled with a same DNS name, either Type-I or Type-II, are chained chronologically, as shown in Fig. 1. For every web server, its continuous history of certificates and publishing keys is archived publicly in the certificate blockchain. We also design a series of regulations, enabling honest web servers to take countermeasures to recover their certificates or publishing keys, after a fraudulent certificate or publishing key is observed.

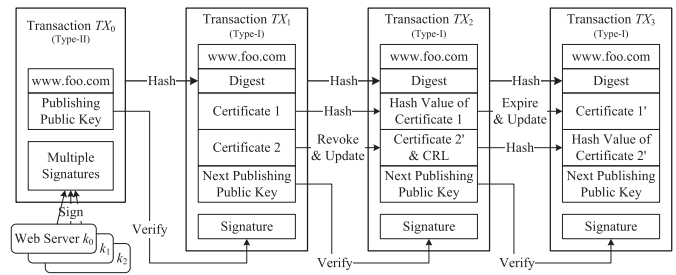


Fig. 1. The history of certificates and publishing keys.

The certificate blockchain is maintained by the miners. A miner collects certificate transactions, verifies the transactions and mines blocks. A browser communicates with the P2P nodes to download the certificate blockchain, and updates its local copy if and only if a longer version is found. When the browser is establishing SSL/TLS sessions with a web server, it validates the server's certificate with the help of its local copy of the certificate blockchain.

3.2 Subject-Controlled Certificate Publication

A certificate is published by its subject, in Type-I certificate transactions. Each Type-I transaction includes:

- 1) *DNS_Name*, the DNS name of the web server.
- 2) *Prev_TX_Hash*, the hash value of the previous transaction with the same DNS name, either Type-I or Type-II.
- 3) *Type*, marked as Type I.
- 4) *Validity*, the validity period of this certificate publication.
- 5) *List_of_Cert_Chain*, a list of published certificate chains. The web server may hold multiple certificates.
- 6) *Next_Publishing_Key*, the publishing public key. The corresponding private key is used to sign the next Type-I transaction with the DNS name.
- 7) *Sig*, the signature of this transaction using the *current* publishing key pair (i.e., the one bound in the most recent transaction with this DNS name, either Type-I or Type-II).

Each transaction is chained to the previous transaction of the same DNS name, by including the hash value of the previous transaction in *Prev_TX_Hash*.

For a certificate chain in *List_of_Cert_Chain*, if it is published for the first time, the whole chain from the root CA certificate to the end-entity certificate is included. It facilitates miners to validate the certificate. When it is published again in the subsequent transactions, only a *simplified entry* is provided, including the hash value of the published certificate and the information to check its "dynamic" validity. Such information includes the validity period, the CRL distribution point or OSCP server location, and the identifier to check its revocation status in CRL files or OSCP responses (e.g., the serial number). The simplified entries enable miners to verify the consistency of these transactions and check the certificate's revocation status, but greatly reduce the overheads of storage and communication.

The web server may update its publishing key pair by itself in *Next_Publishing_Key*, depending on its own security policy. It generates a new publishing key pair and

includes the public key in a Type-I transaction signed by the current key pair. The subsequent Type-I transactions will be signed using the new one.

A certificate is published by its subject. Only the web server that holds the publishing key pair, is able to sign Type-I certificate transactions labeled with its DNS name. A (compromised) CA cannot publish certificates without the consent of the web server.

3.3 CT and RT

A web server periodically publishes its certificates, and all certificates are publicly visible in the inherently append-only blockchain to achieve CT. If a new certificate is signed for the web server, it will be included in the next Type-I transaction, as shown in Fig. 1.

For Type-I transactions, there is an upper limit for the validity period, and it is generally shorter than that of certificates. Thus, a certificate is published for several times during its lifecycle, and then its updated revocation status is also reflected in certificate transactions. When a certificate is revoked and the web server excludes it from the next transaction before it expires, the corresponding revocation information is included instead. So CAs' revocation operations are also recorded publicly in the blockchain. For example, in Fig. 1, a certificate $Cert_2$ is published in the transaction TX_1 and revoked later, so it is excluded from the next transaction TX_2 and the CRL file is in TX_2 instead.

Because the validity period of Type-I transactions shall be greater than the update period of OCSP responses/CRL files, the transparent revocation status may be not so fresh. Anyway, a web server can immediately publish a new Type-I transaction once a certificate is revoked, instead of waiting for the next period.

3.4 Initialization and Reset of Publishing Keys

Type-II transactions are used to *a)* initialize the publishing key of a web server, and *b)* reset it if compromised or lost. Note that, even when the publishing key pair is not compromised, the web server may update it in Type-I transactions. Each Type-II transaction is signed by at least a certain number (denoted as G) of certifier web servers, and also by the certified server itself using a key pair bound in one of its certificates. That is, we require at least G web servers to certify it directly and one CA to do indirectly.

A Type-II transaction includes the following fields:

- 1) `DNS_Name`, the DNS name of the web server.
- 2) `Prev_TX_Hash`, the hash value of the previous transaction with the same DNS name, either Type-I or Type-II.
- 3) `Type`, marked as Type II.
- 4) `Publishing_Key`, the public key of the certified publishing key pair.
- 5) `Certifier_Group`, a list of certifiers' DNS names. The corresponding web servers are authorized to cooperatively control the publishing key of this DNS name.
- 6) `List_of_Cert_Chain`, a list of web server's certificate information, *optional*. It is used to publish certificates as in Type-I transactions.
- 7) `Sig_by_Owner`, a signature by the certified web server. It is verifiable using a CA-signed certificate

binding the DNS name, which is also included in this field.

- 8) `List_of_Sig`, a list of signatures signed by certifiers using their current publishing key pairs. The signers' DNS names are also in this field.

To sign its *initial* Type-II transaction (i.e., the first transaction labeled with the DNS name), a web server contacts at least G servers whose publishing keys have been certified in the blockchain, as its certifiers. This initial transaction is signed by *all* certifiers. An initial Type-II transaction is set with a "frozen" period, before the certified publishing key becomes valid. A typical frozen period lasts for some weeks (e.g., 20 days). A period of some weeks is not considered a large delay in certificate services; for example, it usually takes several days even several weeks to apply an EV SSL certificate. It is worthy to note that this frozen period is necessary only for the *initial* Type-II transactions of a DNS name, but not for any other transactions after the publishing key is initialized. During the frozen period, since a certificate has been signed for the web server by some CA and it is valid in traditional X.509 PKIs, a browser with low security concerns may accept this certificate in SSL/TLS negotiations, but it is still invalid for the browsers supporting our scheme.

The frozen period allows interested parties (not only the web server) to observe squatter attack attempts. If such attack attempt happens, the legitimate owner of the DNS name publishes another initial Type-II transaction during the frozen period, to invalidate the previous certification. Along with the Type-II transaction, the disputer owner will provide extra supporting evidences to claim its ownership. The miners then verify the competitive Type-II transaction and the attached extra evidences, and accept the transaction only if there are more supporting evidences. If such a dispute case happens, the frozen period is automatically extended to allow any other servers to claim the ownership of this DNS name. The publishing key in a Type-II transaction becomes valid, only after the frozen period without any disputes. Note that, only with valid extra evidences, a competitive Type-II transaction will be considered as valid and accepted by miners, and then the frozen period is extended; otherwise, attackers may (collude with a compromised CA to) launch denial-of-service (DoS) attacks, to prevent a legitimate owner's publishing key from becoming valid by competitive initial Type-II transactions.

When miners are verifying an initial Type-II transaction, the typical supporting evidences include: trusted certificates binding the DNS name, and DNSSEC-signed TLSA resource records [39] which list the authorized CA certificates (or public keys) and the web server's certificate (or public key). There are three types of TLS server certificates [82], with different levels of assurance: *a)* domain-validated (DV), only the ownership of the domain name is verified by CAs; *b)* organization-validated (OV), more steps are conducted to confirm the existence of the applicant organization; and *c)* extended-validation (EV), the most efforts are conducted by the CA to validate the certificate request.

Accordingly, when competitive initial Type-II transactions appear, the ranked effectiveness of supporting evidences is as follows: *a)* DV certificates binding the DNS name, *b)* TLSA resource records listing the authorized CA certificates, *c)* OV certificates, *d)* TLSA resource records with the

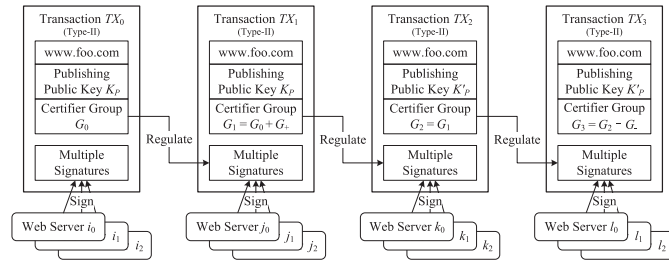


Fig. 2. The certifier group in Type-II transactions.

web server's non-EV certificate, *e*) EV certificates, and *f*) TLSA resource records listing the web server's EV certificate. Meanwhile, any CRL file or OCSP response revoking the certificates which are evidences in the previous competitive transactions, directly destroys the effectiveness of the evidences. That is, the initial Type-II transaction with a TLSA resource record listing the web server's EV certificate, always win the competition, unless this EV certificate is revoked in the evidences of competitive transactions.

For any Type-II transaction, the number of certifiers in *Certifier_Group* must be not less than G . The certifier group can be modified in future Type-II transactions. A *non-initial* Type-II transaction must be signed by G servers in the certifier group specified the previous Type-II transaction, as shown in Fig. 2.

In order to protect publishing keys against compromised certifiers, a series of regulations are enforced on Type-II transactions. So, even if the attackers compromise a trusted CA and enough certifiers, they still need to sign several transactions before successfully publishing a fraudulent certificate. These regulations therefore offer the chances for the web server to recover its certificate or publishing key, before the malicious transactions bring impacts on browsers (see Section 4.1 for details).

Certifiers must be selected carefully, to minimize the possible threat of compromised certifiers. In general, a certifier is a web server that has been introduced into the certificate blockchain without any security incidents for enough time. A server is *not* qualified, if in the past, *a*) one of its certificates was revoked, *b*) its publishing key was reset by Type-II transactions, which implies a key compromise, or *c*) it was removed from some web server's certifier group but not re-introduced soon. If a certifier was innocently removed from a certifier group by attackers, it will be introduced again soon; otherwise, it is considered to be involved in security incidents. Type-I or Type-II transactions automatically contain the details of such incidents. So the miners are able to mark an unqualified server by a special flag in the block header (see Section 3.8), when they are verifying transactions.

When a web server is punished as an unqualified certifier because it was recently removed from some server's certifier group, we do not further punish the certifiers of this unqualified certifier. A chain of propagated punishment inevitably increases the complexity of transaction verification, but does not significantly mitigate the attack impact of malicious or compromised certifiers. In our scheme, in order to be introduced into the certificate blockchain, a web server shall first be certified by a CA. So a group of malicious servers cannot arbitrarily introduce (or certify) any server into the community only by themselves, which will act as another certifier in

the future. At the same time, if a compromised CA colludes with a group of certifiers to introduce a number of servers which then act as certifiers but are involved in incidents, the malicious CA will be detected soon.

The certifier group may be modified after a web server is introduced, but the publishing key and the certifier group cannot be modified *simultaneously* in one transaction, to prevent an attacker from fully controlling a DNS name in the blockchain by only one Type-II transaction. If the attacker compromises G certifiers and modifies the publishing key, the target web server will contact its certifiers to reset the publishing key when it observes the fraudulent transaction. If the attacker modifies the certifier group at first, it cannot immediately control the publishing key to publish fraudulent certificates in the next.

If a group of G_k certifiers is modified in the next Type-II transaction, at most G_- certifiers will be removed and at most G_+ will be introduced, where $G_- \leq G/2$ and $G_+ \leq G/2$ are constants in the system. So it takes more than one fraudulent transactions for an attacker to *a*) deprive the target web server of the control on Type-II transactions by removing its familiar certifiers, or *b*) introduce enough collusive servers as certifiers to sign Type-II transactions arbitrarily.

The optional *List_of_Cert_Chain* field provides an emergency track to *delete* fraudulent certificates published in the previous malicious Type-I transaction. It is used in the extreme attack case that fraudulent certificates have been published by a compromised publishing key, so the target server recovers its control on publishing keys and deletes fraudulent certificates by only one transaction. No revocation information is needed in this field of Type-II transactions. Note that, any certificate appearing in *List_of_Cert_Chain* of Type-II transactions for the *first* time, is considered as *invalid*; otherwise, it provides a fast track for attackers to publish fraudulent certificates.

Sig_by_Owner is a signature by the web server itself. The certificate chain to verify this signature is also included and the certificate chain is signed for the same DNS name. This certificate chain is validated by the certifiers; otherwise, they do not certify the publishing key. Each entry of *List_of_Sig*, contains the DNS name of a certifier and a signature by its publishing key pair. Every certifier uses its currently-valid publishing key to sign this transaction.

3.5 Genesis Block and Founder Servers

In the *genesis* block of the certificate blockchain, $G + 1$ or more special Type-II transactions are included to certify at least $G + 1$ founder web servers' publishing keys. Each of these transactions, is signed by all other founder servers. That is, the publishing keys in the genesis block, are self-certified by the founder servers cooperatively (and some CAs indirectly). The founders shall be highly-ranked web servers. The victims of fraudulent certificates are famous websites [4], [15], [25], [63], [90], so they will be willing to initiate the certificate blockchain to finally tame the absolute authority of CAs in traditional PKIs.

An initial phase is necessary for the founders to invite other highly-ranked web servers (assumed to be honest) to join, i.e., certify other servers' publishing keys. During the initial phase, the certificate blockchain is publicly visible but the community is not open to join. After enough honest servers

are introduced into the community, it is open to the public and the non-founder servers as well operate as certifiers after their publishing keys have been certified in the certificate blockchain.

3.6 Transaction Verification and Block Mining

A miner collects certificate transactions from others and mines the block. Each block consists of a header and multiple transactions, organized as a Merkle hash tree [67].

A block header is composed of: *a*) the time when the miner starts to mine this block, *b*) the digest of the last block header, *c*) the Merkle hash tree root of the included transactions, *d*) a list of (DNS_Name, Type) tuples of the included transactions, lexicographically sorted, and *e*) a PoW nonce computed by brute force, so that the hash value of this block is less than the PoW target. The lexicographical list of (DNS_Name, Type) tuples is included, so that browsers or web servers are allowed to find a transaction without iterating through the included transactions.

Before including transactions into a block and mining the block, a miner verifies their correctness. A Type-I transaction is correct, if and only if:

- 1) Prev_TX_Hash is equal to the hash value of the previous transaction with the same DNS name, either Type-I or Type-II. The publishing key to verify the signature, is bound in that transaction.
- 2) The transaction is in its validity period.
- 3) Each newly-published certificate chain is valid in List_of_Cert_Chain and not revoked under the assumption that the root CA is trusted; and the DNS name appears in CommonName or SubjectAltNames of the certificate [33]. Each simplified entry in List_of_Cert_Chain is consistent with some entry in the last Type-I transaction labeled with the same DNS name, and it does not become expired or revoked.
- 4) The signature of this transaction is correct.

A Type-II transaction is valid to initialize or reset the publishing key of a web server, if and only if:

- 1) Prev_TX_Hash is equal to the hash value of the previous transaction with the same DNS name. If Prev_TX_Hash is zero (i.e., it is an initial transaction), there is no transaction with the same DNS name since the genesis block.
- 2) Publishing_Key and Certifier_Group are not modified simultaneously, compared to the previous transactions with the same DNS name. If Certifier_Group is modified, the publishing key must be identical with the most recent one, in a Type-I or Type-II transaction. If Publishing_Key is reset, the certifier group must be identical with the one in the most recent Type-II transaction.
- 3) If Certifier_Group is modified, at most G_- certifiers are removed and at most G_+ are introduced. An introduced server must be qualified as a certifier.
- 4) The signature in Sig_by_Owner is valid. A miner also validates the certificate in this field under the assumption that the root CA is trusted, and verifies that the certificate is signed for this DNS name.
- 5) If Prev_TX_Hash is not zero, the G signers of List_of_Sig is a subset of the certifier group in the

previous transaction with this DNS name (not the one being verified). If Prev_TX_Hash is zero, the signers compose the certifier group of the transaction being verified.

- 6) The signatures in List_of_Sig are correct. The signatures are verified by the current publishing keys of the certifiers.

In every block, a web server is allowed to publish at most one transaction. If miners receive transactions of different types but with a same DNS name, the Type-II transaction has priority. If there are multiple transactions of the same DNS name and type, a miner selects one of them based on its own policy.

A miner collects mined blocks from other miners, verifies the blocks, and appends valid ones to its local copy to make it longer. A block is valid, if *a*) it is correctly chained to the blockchain, *b*) its hash value is less than the PoW target, and *c*) all included transactions are correct as described above.

3.7 Certificate Validation by Browsers

First of all, a browser communicates with the P2P storage network to incrementally download the up-to-date block headers. Browsers download and store only block headers but no transactions, to reduce the overheads of communication and storage. The browser verifies whether the downloaded headers are chained correctly and each header contains a valid PoW nonce, and updates its local copy if a longer chain is received. This synchronization may be performed when there is no SSL/TLS negotiation.

The certificate transactions to validate a certificate are sent by the visited web server during the SSL/TLS negotiation. A browser sends its certificate transaction request as an SSL/TLS extension. The server, if it supports the proposed scheme, will respond with the transaction as another extension. The identifier of block and the Merkle audit path for the transaction (i.e., the shortest list of additional nodes to compute the Merkle hash tree root [52]) are also sent to enable browsers to verify the certificate transaction.

A certificate chain received in SSL/TLS negotiations is valid, if *a*) it is published in List_of_Cert_Chain of an unexpired transaction, sent by the visited web server, *b*) it is signed by a trusted root CA of the browser, *c*) the transaction is included in a *fully-confirmed* block, i.e., not in the latest N ones of the blockchain, to ensure that the certificate transaction has been accepted by enough miners [7] and the published certificates have been monitored by interested parties, and *d*) in the blocks subsequent to this fully-confirmed block, including the N non-fully-confirmed ones, if there is any transaction with the visited DNS name, then the certificate shall also appear in the transaction.¹

Because the certificate chain has been validated by the majority of honest miners, the browser only needs to check whether the root CA certificate is trusted by itself or not. Other processing such as CA signatures, periods of validity, revocation status and certificate extensions [16], is unnecessary. That is, browsers delegate the certificate validation to

1. The additional transactions are also sent by the web server. This regulation is designed to deal with revoked or fraudulent certificates (see Section 4.1). For Type-II transactions, it takes effect only if the optional List_of_Cert_Chain field appears.

the community of web servers, and the delegated certificate validation is also transparent (i.e., publicly visible).

Finally, since the validity period of Type-I transactions shall be greater than the general update period of OCSP responses/CRL files and then the revocation status in certificate transactions may be not so fresh, a browser with high security concerns may take extra actions to check the revocation status.

3.8 Limited Storage of Blocks

Expired Type-I transactions are useless in certificate validation, so a browser only needs to store the recent block headers within the upper limit of the validity period of Type-I transactions (denote as T_I). T_I is set great enough (e.g., 10 days), so the longest branch of the blockchain is always mined on one of the stored headers.

A miner needs to store all blocks, including headers and transactions, to *a)* verify the relationship of certificates and publishing keys of a web server among its Type-I and Type-II transactions, *b)* find the certifiers' publishing keys to verify Type-II transactions, and *c)* check whether a server is qualified as a certifier or not.

The following designs reduce the storage requirement of miners. Each web server publishes its Type-II transactions *periodically*, even when it does not reset the publishing key pair or modify its certifier group. The period is denoted as T_{II} and $T_{II} > T_I$. In these "shadow" Type-II transactions, both `Publishing_Key` and `Certifier_Group` must be identical with those in the previous transaction, and `List_of_Cert_Chain` must be absent. A shadow transaction is signed only by the web server itself, and no signature by certifiers is needed. Finally, if a web server is involved in any incident, a flag is set with its DNS name in the block header.

So a miner only stores *a)* the recent block headers within the certain period of time (denoted as T_G , and $T_G > T_{II}$), to check whether a server is qualified as a certifier, and *b)* the latest transactions of both types of each DNS name within T_{II} . If a web server does not publish shadow Type-II transactions, it is still able to publish periodical Type-I transactions, but cannot reset its publishing key or modify the certifier group any more.

3.9 Mining Incentives

We design the incentive mechanism for the block mining, to keep difficulties for attackers. Like other crypto-currencies, a number of cert-coins are generated as a block is mined and rewarded to the miner. Every Type-I transaction published in the blockchain consumes one cert-coin. Cert-coins are traded and transferred among servers, so a web server may purchase cert-coins from others to publish its certificates.

We do not design a general-purpose crypto-currency, and cert-coins are designed specially for certificate transactions. When a block including R Type-I transactions is mined, $f \times R$ cert-coins are generated and rewarded to the miner, where the reward coefficient f is a system constant and $f > 1$. A cert-coin becomes invalid *automatically*, after a certain period of time (denoted as T_C) since it was generated. In a short-term view, the valid cert-coins are always a little more than those consumed in certificate transactions, which promotes the trades among servers; while in a long-term

view, the number of cert-coins does not inflate. We let $T_C \leq T_{II}$ and then miners only need to keep the recent transactions within T_{II} to verify cert-coins.

The identity of the miner is contained in the block header, for the cert-coin reward. A `Cert_Coin` field is included in every Type-I transaction to describe the consumed cert-coin, and the miners check whether it is valid or not. Finally, a cert-coin transaction is signed by the owner, to transfer cert-coins to other servers, also verified by miners.

Therefore, compared with other PoW-based cryptocurrencies (e.g., Bitcoin, ETH and Litecoin), the verification during the mining of cert-coins is improved, while the PoW computations are identical. All data verified in the mining have validity periods, so that the storage requirement and verification complexity do not increase remarkably as time goes by. In particular, only the recent block headers within T_G , the transactions within T_{II} , and the cert-coins generated with T_C where $T_G > T_{II} > T_I$ and $T_C \leq T_{II}$ are needed during the mining of our scheme.

4 SECURITY ANALYSIS

If CAs are trustworthy and web servers are honest, a web server periodically publishes its certificates using the publishing key pair, which is certified by at least G servers. If a certificate becomes expired or revoked, it will be excluded. Meanwhile, a web server regularly updates its publishing key, after it is introduced into the community (i.e., its publishing key is initially certified). If the publishing key is lost, the web server will contact its certifiers to reset it.

Next, the proposed scheme is analyzed under various attack scenarios as follows. When some entities except the target web server were compromised, we present the corresponding countermeasures and evaluate the attack impacts. In this attack analysis on certificate transactions, we assume that the certificate blockchain is append-only, so that a fully-confirmed transaction never will never be deleted or modified. Then, the network attacks on the certificate blockchain are also analyzed. Note that, an attack is successful, if and only if a fraudulent certificate is published in the certificate blockchain and accepted by some browsers.

4.1 Security With Compromised Key Pairs

In order to impersonate a web server, an attacker might compromise a CA to sign fraudulent certificates, or compromise some web servers' publishing key pairs. We do not consider the situation that, the key pair bound in the target server's certificate is compromised by attackers. In this situation, the attackers could directly impersonate the web server and launch MitM attacks, without fraudulent certificates. Such attacks should be prevented or mitigated by the approaches other than certificate management, which are out of the scope of this paper.

Table 1 enumerates all possible subsequent attack actions when different key pairs are compromised. As shown in the table, if only publishing key pairs are compromised, the attackers can set `List_of_Cert_Chain` or modify the publishing key pair in fraudulent Type-II transactions. When the attackers compromise CAs and some servers' publishing key pairs, the possible subsequent attack actions include: publish fraudulent certificates, modify the publishing key pair of the

TABLE 1
The Possible Attack Actions on Certificate Transactions

Compromised Entity or Component	Possible Subsequent Attack Action	
Only CAs	No attack impact; see Section 4.1.1.	
Only publishing key pairs	Set <code>List_of_Cert_Chain</code> , or modify the publishing key pair. Section 4.1.2.	
CAs and publishing key pairs	Publish fraudulent certificates.	Section 4.1.3.
	Modify the publishing key pair.	Section 4.1.3.
	Compromise the publishing key pairs of G certifiers.	Set <code>List_of_Cert_Chain</code> . Section 4.1.3.A.
		Reset the publishing key pair. Section 4.1.3.B.
		Modify the certifier group. Section 4.1.3.C.
		Publish fraudulent certificates. Section 4.1.3.D.

target server, and compromise the publishing key pairs of G certifiers. Then, if the publishing key pairs of G certifiers are compromised, the possible attack actions include: set `List_of_Cert_Chain`, publish fraudulent certificates, reset the publishing key pair, and modify the certifier group.

In the next sections, we present the countermeasures taken by the honest web servers after the fraudulent transactions are observed, and the attack impacts are discussed one by one. To facilitate the description, we analyze the action of fraudulent certificates after G compromised publishing key pairs of certifiers in the last paragraphs (i.e., Section 4.1.3.D).

In the discuss of countermeasure transactions and attack impacts, we always assume that the certificate blockchain is *append-only*, so that honest web servers can recover their certificates or publishing keys only by countermeasure transactions after a fraudulent transaction is included in the blockchain, but not deleting or modifying this fraudulent one. At the same time, the attackers cannot modify a confirmed transaction in the certificate blockchain, even after the key pairs of all signers of the transaction are compromised, and then it offers the chances for honest web servers to counter the impact of fraudulent transactions.

4.1.1 Compromised CAs

When only CAs are compromised, our scheme ensures that no fraudulent certificate is accepted by browsers, because the target web server will not publish such certificates.

4.1.2 Compromised Publishing Key Pairs

An attacker is unable to impersonate the target web server, if it only compromises publishing key pairs but not any CAs. The attacker might modify `List_of_Cert_Chain` in a Type-I transaction, by including an expired or revoked certificate, or excluding a currently-valid one. But the transaction verification will fail, because miners will validate certificates and only expired or revoked ones shall be excluded compared with the previous transactions.

The attacker might modify `Next_Publishing_Key` with another key pair through Type-I transactions, to prevent the target server from updating its Type-I transactions. The web server will reset its publishing key by a Type-II transaction, with the cooperation of G certifiers. Note that,

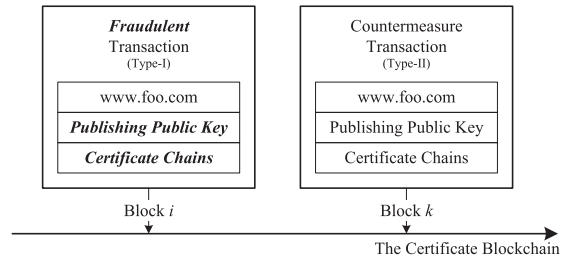


Fig. 3. A fraudulent Type-I transaction in Block i , and the countermeasure Type-II transaction in Block k . If $i < k \leq i + N$, the fraudulent certificate is never accepted; otherwise, if $k > i + N$, it may be accepted before the countermeasure transaction.

in this case, the attacker cannot sign fraudulent Type-II transactions, which requires a signature verified using the server's CA-signed certificate.

4.1.3 Compromised CAs and Publishing Key Pairs

Attackers might compromise CAs to sign fraudulent certificates and also compromise some server's publishing key pair. In the following, we assume that, for the target web server, in the certifier group of its most recent non-fraudulent Type-II transaction, there are at least G_h ($G_h \geq G$) honest server. Note that, the honest certifier might be removed from `Certifier_Group` in subsequent fraudulent Type-II transactions, and compromised servers might be introduced into the certifier group. The total number of compromised servers could be greater than G finally.

First of all, if the target server's publishing key pair is compromised, the attackers could publish fraudulent certificates in a Type-I transaction, and simultaneously modify `Next_Publishing_Key` with another key pair held by itself. This transaction will be verified by miners, and included into a mined block.

Once the fraudulent Type-I transaction is observed by the target server or any other interested parties, the server will contact G honest certifiers to sign a countermeasure Type-II transaction, to exclude the fraudulent certificates by properly setting `List_of_Cert_Chain` and simultaneously reset its publishing key in `Publishing_Key`.

As shown in Fig. 3, if the countermeasure transaction appears in time (i.e., in any of the N subsequent blocks after the fraudulent transaction), browsers will detect the conflict and reject the fraudulent certificates (see Section 3.7). Such a Type-II transaction thoroughly counters the impact of the fraudulent Type-I transaction. If the countermeasure transaction is not signed in time, the fraudulent certificates might be accepted temporarily but rejected after the countermeasure transaction.

Alternatively, the attackers might compromise the publishing key pairs of the target web server's certifiers, and then take actions through fraudulent Type-II transactions. After compromising the publishing key pairs of G certifiers and colluding with a CA, the attackers could sign a Type-II transaction which will be included in the blockchain. Different possible subsequent attack actions are analyzed as follows.

A. Set `List_of_Cert_Chain` in Type-II transactions

By setting `List_of_Cert_Chain` in the fraudulent Type-II transaction, the attackers exclude some valid certificates

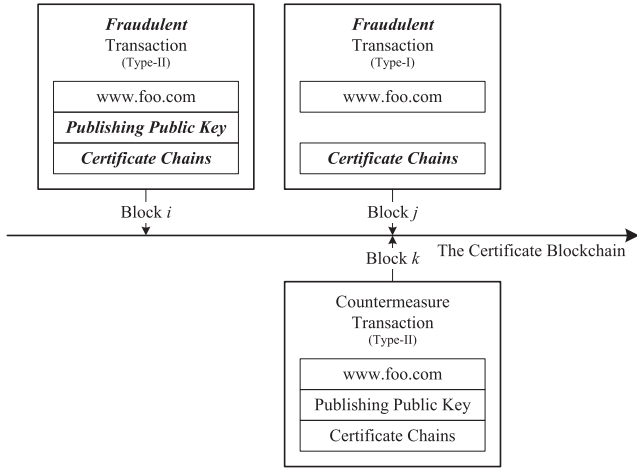


Fig. 4. A fraudulent Type-II transaction in Block i , and the countermeasure Type-II transaction in Block k . If $i + 1 < k$, a fraudulent Type-I transaction may be included in Block j ($i < j < k$); otherwise, no fraudulent certificate is published. If $k \leq j + N$, the fraudulent certificate is never accepted; otherwise, if $k > j + N$, it may be accepted.

without revocation information. It leads to DoS attacks and valid certificates will be rejected by browsers, but no fraudulent certificate is accepted. The attack impacts will be eliminated, after the target web server publishes another Type-I transaction for the valid certificates again.

B. Reset the publishing key in Type-II transactions

Attackers might reset the target server's publishing key to another one held by himself, by a fraudulent Type-II transaction. Then, in next blocks, it attempts to publish fraudulent certificates in Type-I transactions. Note that, in one block, multiple transactions are prohibited to be labeled with a same DNS name.

Once the target web server observes the fraudulent transactions, it will immediately contact G certifiers to reset its publishing key by another Type-II transaction. This countermeasure transaction will reset its publishing key and also include all its valid certificates in `List_of_Cert_Chain`, whether the subsequent fraudulent Type-I transactions have been signed or not, as shown in Fig. 4. The countermeasure workload is the same as that of resetting its publishing key in the case of fraudulent Type-I transactions in Section 4.1.2, because the attackers cannot simultaneously modify the certifier group.

Moreover, the countermeasure Type-II transaction has a greater probability to be included in the blockchain, than the fraudulent Type-I transaction, because Type-II transactions have priority during the block mining (see Section 3.6). After the countermeasure Type-II transaction has been included into the blockchain, the fraudulent Type-I transactions become incorrect because the publishing key has been reset. Again, only if the countermeasure Type-II transaction is included in the blockchain after the fraudulent Type-I transaction has been included and N more blocks have been mined, a browser may accept the fraudulent certificates.

After observing such a fraudulent Type-II transaction, the target server also notifies the signers that their publishing key pairs are compromised. These certifiers will also start to recover their publishing keys, as the target server.

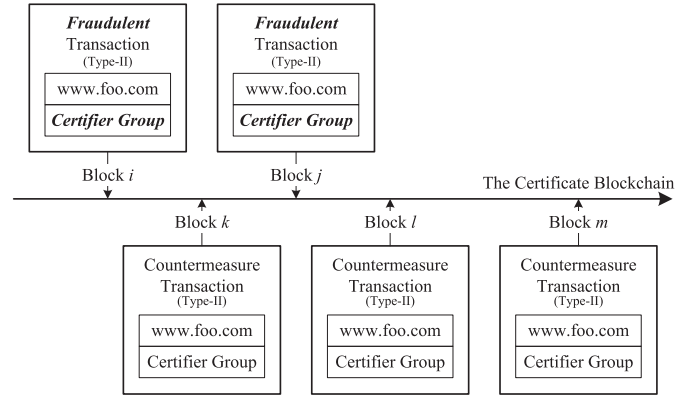


Fig. 5. Fraudulent Type-II transactions in Blocks i and j to control the certifier group, and the countermeasure transactions in Blocks k , l , and m to compete for the control.

Once enough publishing key pairs are recovered, the attack is prevented thoroughly. If some certifiers are detected to collude with (or be compromised by) the attackers, they will be removed from the certifier group and considered as an unqualified certifier for a period of T_G .

C. Modify the certifier group in Type-II transactions

Attackers might choose to modify the target server's certifier group. If all servers in the certifier group are unfamiliar to the web server (and the publishing key is modified), it becomes very difficult for the target web server to re-control its publishing key pair.

Once the target web server observes the fraudulent Type-II transaction that modifies its certifier group, it will immediately contact its certifiers to sign countermeasure Type-II transactions. The target web server will *a*) introduce or re-introduce familiar web servers as certifiers, *b*) notify the signers of the fraudulent transaction that their publishing keys have been compromised, and *c*) remove the certifiers introduced or compromised by the attackers.

Then, the target server and the attackers will compete for the control on the certifier group, by introducing/removing certifiers, as shown in Fig. 5. In order to ensure that the attack will finally fail, if the target web server takes countermeasures in the n th block after the fraudulent transaction, the following equations shall be satisfied: $G_h - n \times G_- \geq G$ and $n \times G_+ < G$.

If $G_h - n \times G_- \geq G$ and $n \times G_+ \geq G$, the target server and the attackers will compete for the control on the DNS name, and the result depends on the policies of miners. If $G_h - n \times G_- < G$ and $n \times G_+ \geq G$, the attack will succeed. As an extension, a web server is allowed to customize its G_+ and G_- in Type-II transactions (e.g., smaller G_+ and G_-), based on its frequency to monitor the certificate blockchain.

Because the publishing key and the certifier group cannot be modified simultaneously in one Type-II transaction, the web server is still able to use the unmodified publishing key pair to sign Type-I transactions regularly.

D. Sign fraudulent Type-I transactions, and modify the certifier group through Type-II transactions

The attackers may first publish fraudulent certificates in a Type-I transaction. Then, they attempt to modify the certifier

TABLE 2
The Countermeasure Transactions for Different Attack Scenarios

Scenario	Relationship and Note
4.1.1 & 4.1.2	No attack impact.
4.1.3.-	If $n \leq N$, no attack impact.
4.1.3.A	It results in DoS attacks, before the countermeasure transaction is fully-confirmed.
4.1.3.B	If $n \leq N + 1$, no attack impact.
4.1.3.C	If $G_h - n \times G_- \geq G$ and $n \times G_+ < G$, no attack impact.
4.1.3.D	If $n \leq N$, $G_h - (n - 1) \times G_- \geq G$ and $(n - 1) \times G_+ < G$, no attack impact.

group by fraudulent Type-II transactions. In this case, the target web server will contact its certifiers to sign Type-II transactions, as in the above case to compete for the control on the certifier group. At the same time, the countermeasure Type-II transactions will exclude the fraudulent certificates by properly setting `List_of_Cert_Chain`.

4.1.4 Summary

The target web server depends on Type-II transactions to counter fraudulent transactions, either Type-I or Type-II. When fraudulent certificates have been published, they are excluded by properly setting `List_of_Cert_Chain` that lists valid certificates only. If the attackers start to modify the certifier group, the countermeasure Type-II transactions compete for the control on the certifier group; otherwise, the countermeasure transactions reset the publishing key.

Table 2 lists the countermeasure transactions, after the first fraudulent transaction in Block i . The first countermeasure transaction is in Block k ($k > i$, and $n = k - i$). In order to ensure that no fraudulent certificate is accepted by browsers, it requires that $n \leq N$, $G_h - n \times G_- \geq G$ and $n \times G_+ < G$. A web server will find its tradeoff based on *a*) the frequency to monitor the certificate blockchain, *b*) the honest certifiers that it communicates with, and *c*) the customized G_+ and G_- .

If the web server finally loses the control on its DNS name in the competition, *out-of-band* actions are necessary to recover the ownership in the certificate blockchain. It has to publish another initial Type-II transaction, and takes *out-of-band* actions to convince the majority of miners (not only its certifiers) to accept this transaction, because the new initial transaction violates the previous records in the blockchain.

4.2 Attacks on the Certificate Blockchain

In general, there are potential attacks on blockchains or Bitcoin, such as eclipse attacks [37], double-spending attacks [72], selfish-mining attacks [29], [74], and block-withholding attacks [17], [71]. Next, we discuss these attacks, when they are launched on the certificate blockchain.

In particular, the eclipse attackers [37] might attempt to prevent browsers from accessing the certificate blockchain to download the recent block headers. If such an attack is performed, the victim browser is aware of it because the mining time is contained in block headers. In fact, it is extremely difficult to isolate a browser from a great number of P2P nodes, while allow it to access (counterfeit) web servers at the same time. We can further defeat this attack by instructing the

browser to require the recent block headers from the visited web server, if it does not store these headers.

Malicious miners might accept invalid transactions arbitrarily when mining blocks, and broadcast them to other entities [72]; e.g., a Type-I transaction publishing fraudulent certificates but with an invalid signature, or two conflicting Type-II transactions one of which modifies the publishing key and the other modifies the certifier group. Since the majority of nodes are honest, it is practically impossible that invalid transactions are included in any longest branch of the certificate blockchain.

Or, the attackers might collude with compromised CAs and certifiers, to sign a fraudulent certificate and include it in a manipulated branch of the blockchain [72]. Then, subsequent blocks are mined on this branch, while countermeasure transactions are elaborately excluded. This attack violates the assumption that honest entities communicate with each other in a loosely synchronized manner. In other words, the certificate services do not require the status to be confirmed quickly (e.g., the typically validity period of CRL files is 7 days and the period of OCSP responses is 4 days, according to the statistics in the Internet in 2018 [56]), so some attacks against blockchains in other online applications are ineffective in our scheme. In particular, during the frozen period the attackers might attempt to *temporarily* manipulate a longer branch excluding the competitive Type-II transaction; however, because the frozen period typically lasts for some weeks, such attacks are practically impossible in our scheme. Or, if some fraudulent transaction is included in the blockchain but not fully-confirmed in a temporary branch, this transaction may lead to some valid certificates to be rejected but no fraudulent certificate is accepted by browsers (see Sections 3.7 and 4.1.3 for details).

Other attacks on mining pools [17], [29], [71], [74] influence the mining incentives (or cert-coins) allocated among miners, but they do not prevent valid transactions from being included in the mined blocks. We assume that at least 67 percent of computation resources are controlled by honest web servers [29], so the attackers cannot manipulate the blocks in the long term.

Finally, there are also attacks against user privacy in the blockchain or Bitcoin [47], [80]. Such attacks are irrelevant to our scheme, because there is no privacy information in the data of the certificate blockchain.

5 IMPLEMENTATION AND EVALUATION

In this section, we discuss the parameters of our scheme, present the prototype and evaluate its performance.

5.1 Parameters

The time interval between two adjacent blocks (denoted as T_B) determines how soon a certificate will be accepted by browsers after it has been included in the blockchain. It is reasonable for a web server to require its published certificates to be accepted within 24 hours, i.e., $N \times T_B < 1,440$ minutes. On the other hand, a smaller T_B enforces the web server to monitor the certificate blockchain more frequently, and take countermeasures more quickly. We set $T_B = 120$ minutes as a typical value and let $N = 6$ (the same as the requirement in Bitcoin [7]).

TABLE 3
Parameters of the Proposed Scheme

Notation	Value	Description
N	6	The number of unconfirmed blocks in the blockchain.
G	5 or 10	The least number of certifiers to sign Type-II transactions.
T_B	120min	The time interval between two blocks.
T_I	10d	The period of Type-I transactions.
T_{II}	100d	The period of shadow Type-II transactions.
T_C	100d	The lifetime of cert-coins.
T_G	400d	The penalty period of unqualified certifiers.

The validity period of Type-I transactions (denote as T_I) is chosen to introduce incentives for web servers to mine blocks. Only when a transaction is included in a fully-confirmed block (not in the latest N ones of the blockchain), the contained certificates are considered as valid by browsers. So $T_I \gg (N + 1) \times T_B$; otherwise, it is never accepted by browsers before it expires.

T_I shall be not significantly greater than the general revocation status update period, to enforce the web servers to update their transactions in a timely manner and provide RT. So we require that $T_I \leq 10 \times T_{Revoke}$, where T_{Revoke} is the typical revocation status update period. Although most CAs are able to update the CRL files within 24 hours [94], the typically validity period of CRL files is 7 days [56]. The validity period of OCSP responses is typically 4 or 7 days.² Thus, we set $T_I = 14,400$ minutes (or 10 days) in the prototype.

T_{II} determines the frequency of shadow Type-II transactions in the certificate blockchain. We set $T_{II} = 10 \times T_I$ (i.e., 100 days), $T_C = T_{II}$, and $T_G = 4 \times T_{II}$ in the prototype. We evaluate the performance, when $G = 5$ or 10, i.e., at least five or ten certifiers are required to sign a Type-II transaction. Table 3 lists parameters of the proposed scheme.

5.2 Implementation and Experiment Setting

The prototype system is composed of, *a*) a browser that validates certificates, based on its local copy of the certificate blockchain, *b*) a web server that delivers its certificates transactions as SSL/TLS extensions, and *c*) an instance of the certificate blockchain.

The browser is implemented on Firefox Nightly v54.0a1. The function `VerifySSLServerCert()` is modified as described in Section 3.7, to validate a certificate based on the certificate blockchain (and its root CA setting). The web server is implemented on Nginx (version 1.10.3). Extensions of `ClientHello` and `ServerHello` are defined and implemented in the SSL/TLS handshake, to request and send the certificate transactions and the corresponding Merkle audit paths. The browser and the web server are connected directly. The browser runs on a desktop (Intel i7-4770s/3.10 GHz CPU, 8 GB RAM, and ST-1000DM003 hard disk) with Windows 7, and Nginx is on another desktop of the same hardware configuration with Ubuntu 16.04 LTS (32-bit).

2. We visited the Alexa top-50 websites in 2017, and found 29 unique certificate chains (averagely 4.05 KB), each of which is composed of three certificates. We collected OCSP responses (averagely 1.60 KB) for the server certificates, and the distribution of validity periods is: 17 are 7-day, 9 are 4-day, 2 are 1.5-day, and 1 is 5-day.

There are 54.35M SSL/TLS certificates in 2016 [10], and each website owns 1.34 certificates on average [89]. However, nearly 70 percent are issued for free (e.g., by Let's Encrypt or cPanel), and many are signed for phishing websites [23] or even inaccessible [89]. The free certificates provide little trust for authentication, and shall not be included in the interdependent community. So we focus on 13.88M non-free certificates, corresponding to about $13.88M / 1.34 = 10.36M$ websites.

We implement the miners based on OpenSSL-1.0.1g. It verifies the transactions as described in Section 3.6. However, we do not compute the PoW nonce and it is not verified in the experiment. In the experiment, it runs on a desktop of the same configuration with the browser. The prototype certificate blockchain for 10.36M websites, contains 1,200 blocks within T_{II} and transactions for 10.36M websites. In the prototype certificate blockchain, each server publishes one Type-II transaction and averagely $1200/114 = 10.5$ Type-I transactions (i.e., one Type-I transaction every 114 blocks, to ensure that a transaction does not expire until the next one is fully-confirmed). So on average each block contains $10.36M/114 \approx 93.06K$ Type-I transactions and $10.36M/1200 \approx 8.84K$ Type-II transactions. 34 percent of Type-I transaction includes two certificate chains, and others contain one chain; i.e., averagely 1.34 certificates per website. Each chain is composed of three average-size certificates. SHA-256 and RSA-2048 are adopted, which is the common practice in the real-world community. DNS names are randomly generated with the average length of 14 bytes, which is the average length of Alexa top-1M websites in 2017.

OCSP Stapling messages are included as the revocation status information, for it is widely supported in current PKI systems [57] and its size (1.60 KB on average, including the OCSP responder's certificate) is smaller than that of a CRL file (averagely 51 KB [57]). In the prototype certificate blockchain, 1 percent of certificates are revoked according to the real-world statistics [57]. Meanwhile, the revocation rate may increase to 11 percent when a critical incident happens [57].

5.3 Performance Evaluation

The scheme is evaluated in terms of *a*) the storage requirement of browsers and web servers, *b*) the certificate validation delay of browsers, *c*) the communication between browsers and web servers, *d*) the transaction verification by miners, and *e*) the cost of web servers to publish certificates.

5.3.1 Storage

Table 4 lists the average size of each element. A browser keeps the block headers within T_I , so the storage overhead is about $120 \times 1.40 \text{ MB} \approx 168.00 \text{ MB}$.

Each miner keeps the latest Type-I and Type-II transactions of each DNS name, and all block headers within T_G , which is $10.36M \times (0.91 \text{ KB} + 1.02 \text{ KB}) + 1200 \times 4 \times 1.40 \text{ MB} \approx 26.64 \text{ GB}$. All transactions in the N non-fully-confirmed blocks shall be kept always, to ensure the verification of blocks in multiple concurrent non-fully-confirmed branches; and the size is $6 \times (93.06K \times 0.91 \text{ KB} + 8.84K \times 1.02 \text{ KB}) \approx 0.55 \text{ GB}$. That is, a miner stores at least 27.19 GB data. In the typical case that 1 percent of certificates are revoked by OCSP, the storage overhead increases to 27.36 GB; if 11 percent are revoked by CRL, it is 109.62 GB. In fact, even if the blockchain includes all

TABLE 4
The Size of Data Elements in the Blockchain

Item	Average Size
<i>Certificate chain, with 3 certificates</i>	4.05 KB
<i>Simplified certificate chain entry</i>	240B
<i>OCSP Stapling message, with 1 certificate</i>	1.60 KB
<i>CRL file</i>	51.00 KB
<i>Type-I transaction</i>	
with 1/2/3 certificate chain	4.65/8.71/12.77 KB
with 1/2/3 simplified entry	0.83/1.07/1.30 KB
<i>Type-II certificate transaction</i>	
without certifier signature (shadow)	1.02 KB
with 5/10 certifiers' signatures	5.10/5.36 KB
<i>Block header</i>	1.40 MB
<i>Block, including the header and transactions</i>	
no certificate revoked	95.05 MB
1% revoked & updated, OCSP/CRL	101.05/162.62 MB
11% revoked & updated, OCSP/CRL	161.13/838.42 MB

54.35M certificates, the typical storage overhead of a miner will not exceed 150 GB, which is the number for a miner of Bitcoin.

5.3.2 Certificate Validation Delay

A browser validates the received certificate chain as follows: *a)* checks whether the root CA is trusted, *b)* reads a block header from the hard drive, *c)* checks if a block header contains a transaction of the visited DNS name, and *d)* checks whether the received transaction is in the found block and the certificate is in the transaction. Operations *b)*, *c)* and *d)* are performed repeatedly, until it confirms that the certificate is in an unexpired transaction in the recent fully-confirmed blocks (and also all of its subsequent transactions in non-fully-confirmed blocks, if appear).

The browser validates certificates, when the target transaction is in the middle/end of the local copy of the certificate blockchain (i.e., the average/worst case, 1,000 times for each case). The average delay is listed in Table 5. The time cost is greatly reduced if a browser loads all block headers (about 168.00 MB) into memory when it starts up; similarly, Firefox preloads all trusted root CA certificates into memory when it starts up to accelerate certificate validation. In such case, Operation *b)* is unnecessary and the delay is 0.98 ms. This number will increase to 3.20 ms if the blockchain includes 54.35M certificates. On the contrary, the browser takes about

TABLE 5
Certificate Validation in Browsers

Operation	Average Time
<i>Standard certificate validation</i>	9.86 ms
<i>Basic operation</i>	
<i>a) Check trust anchors</i>	< 0.01 ms
<i>b) Read a block header from hard drive</i>	1.11 ms
<i>c) Search the DNS name in a block header</i>	0.01 ms
<i>d) Check a transaction and the certificate</i>	0.25 ms
<i>Validation with the certificate blockchain</i>	
Operations <i>a)+b)+c)+d)</i> , in the middle/end	71.87/135.38 ms
Operations <i>a)+c)+d)</i> , in the middle/end	0.98/1.65 ms

TABLE 6
The Transaction Verification by Miners

Item	Average Time
<i>Block header</i>	7.03s
<i>Type-I transaction</i>	
with 1 certificate chain, by OCSP/CRL	13.67/11.69 ms
with 1 simplified entry, by OCSP/CRL	4.01/1.83 ms
with 1 certificate chain, 1 simplified entry, and 1 revoked & updated certificate chain, by OCSP/CRL	17.57/14.42 ms
<i>Type-II transaction</i>	
with 10 certifiers' signatures (initial)	12.33 ms
with 5-out-of-10 certifiers' signatures	11.44 ms
with no certifier signature (shadow)	0.30 ms
<i>Block</i>	
no certificate revoked, by OCSP/CRL	1023.67/467.16s
1% revoked & updated, by OCSP/CRL	1035.11/480.50s
11% revoked & updated, by OCSP/CRL	1233.46/608.20s

9.86 ms to validate a certificate, using the standard certificate validation except checking its revocation status.

5.3.3 Transaction Verification by Miners

Before including transactions into a block and mining the block, a miner verifies the correctness of all Type-I and Type-II transactions. Or, on receiving a block, the miner needs to verify the block header and also all included transactions.

Table 6 lists the experimental results. First, it takes on average 7.03 s to verify a block header, including the following fields: *a)* the digest of the last block header, *b)* the Merkle hash tree of all included transactions, *c)* the list of (DNS_Name, Type) tuples, and *d)* the PoW nonce.

For each Type-I transaction, the miner needs to verify all fields, mainly including the list of published certificate chains and the signature of this transaction. When only one certificate chain is published, it takes 13.67/11.69 ms to verify the transaction, including the time cost to check the revocation status by OCSP responses/CRL files. Note that, the time to download OCSP responses/CRL files is not counted. Meanwhile, it takes 4.01/1.83 ms to verify a Type-I transaction with one simplified entry. When the transaction includes a certificate chain, a simplified entry, and a revoked & updated certificate chain, it takes 17.57/14.42 ms.

The results of different Type-II transactions are as follows. It takes on average 12.33 ms to verify an initial Type-II transaction with 10 certifiers. For non-initial Type-II transactions, it takes 11.44 ms to verify a Type-II transaction signed by 5 certifiers, and only 0.30 ms to verify a shadow Type-II transaction with no signature by certifiers.

Finally, in the prototype certificate chain, if no certificate is revoked, it takes on average 1023.67 and 467.16 s for the miner to verify a block, including the verification of all transactions, when the revocation status is checked by OCSP responses and CRL files, respectively. When 11 percent of certificates are revoked and updated, the time cost to verify a block increases to 1233.46/608.20 s.

5.3.4 Communication

In an SSL/TLS negotiation, the introduced communication overhead is typically one certificate transaction between the

browser and the web server, or $N + 1$ transactions in the worst case. It is about 1.53 KB, or 10.71 KB (when $N = 6$). In the worst case, one transaction is in a fully-confirmed block, and six transactions in non-fully-confirmed blocks.

A browser incrementally updates its local copy of block headers, about 16.80 MB every day. The headers are reduced to about 5~8 MB by the ZIP compress, because the lexicographically-sorted DNS names of web servers occupy the most size of a block header.

5.3.5 Transaction Cost and Mining Incentive

Enough incentives for block mining are necessary to keep difficulties for the attacks directly on the blockchain. Next, we analyze the incentives to draw computation resources comparable to those of typical blockchain-based crypto-currencies. The overall computation power of Bitcoin is 4,788 PHash/s, and an AntMiner S9-13.5T of 13,500 GHash/s earns 8.81 USD per day [79]. Assume that the certificate blockchain has the same computation power as Bitcoin, and then such a mining machine generates $(13,500\text{G}/4,788\text{P}) \times (24\text{h}/T_B) \times f \times 93.06\text{K} \approx 3.69$ cert-coins per day ($f = 1.2$). Therefore, if the cert-coin price is greater than $8.81/3.69 = 2.39$ USD, miners are willing to invest in the certificate blockchain. The computation power of ETH (or Litecoin) is 27THash/s (or 6542 GHash/s), and the power of a typical mining machine is 230 MHash/s (or 275 MHash/s) [79], [88]. So the cert-coin price is 1.88 and 0.23 USD, if ETH and Litecoin are taken as the reference, respectively.

So a web server will pay about 87.24/68.62/8.40 USD per year, for the same level of mining incentive as Bitcoin/ETH/Litecoin, which is not greater than the regular certificate price. Note that a DV certificate costs about 64 USD for one year, and an EV certificate costs 199 USD [14].

Next, we discuss how the system parameters influence the cost to publish certificate transactions (or the mining incentives). The cert-coin price with enough incentives to keep the same computation power as Bitcoin is $8.81/M$, where $M = f \times (13,500\text{G}/4,788\text{P}) \times (24\text{h}/T_B) \times 10.36\text{M}/(T_I/T_B - N)$. Since $N \ll T_I/T_B$, the cert-coin price is approximately proportional to $T_I/(f \times 10.36\text{M})$. Here, 10.36M is the number of websites in the Internet. A web server pays one cert-coin for each Type-I transaction, and then *a*) if more websites participate in the community and pay for the mining, each web server pays less for its Type-I certificate transaction; *b*) if T_I becomes greater, a smaller number of Type-I transactions are included in the certificate blockchain and the demand of cert-coins becomes smaller; and *c*) when the reward coefficient f is greater, more cert-coins are generated in the mining and the inflation leads to cheaper cert-coins.

5.4 Comparison With the Certification Solutions on Top of Blockchains

There are schemes to manage identities and public keys (or certificates) in blockchains. We compare them with our scheme as follows.

Namecoin [68] is a decentralized blockchain-based DNS system, associating DNS names with Bitcoin addresses (or key pairs). Crypto-currencies are generated as blocks are mined, and used to register .bit DNS names. [31], [32] extends Namecoin by associating two key pairs with a DNS

name. Then, the online key pair is used in SSL/TLS negotiations and the offline one is used to revoke and update the online one. In these systems, the key pairs are controlled entirely by the owner of the DNS name. If the private key is compromised, the attacker will associate any key pairs with the DNS name and such intrusions cannot be recovered. Namecoin and its extensions allow the owner of the DNS name to manage and certify its key pair by itself in the public blockchain, but not by any central authorities. That is, compared with traditional PKIs, these schemes shift the trust from CAs to the owner of the DNS name. On the contrary, our solution distributes the control among CAs and the community of web servers (i.e., the trust on CAs is extended to both the CAs and the community of web servers), so in our scheme an attacker cannot bind a key pair to the DNS name after breaking only a single entity.

Blockchain PKIs [53] store credentials in the Ethereum blockchain (i.e., a collection of key-value pairs controlled by a CA), and revoked credentials in another on-chain storage also controlled by the CA. CertChain [12] proposes to record CAs' certificate signing and revocation operations in two separate blockchains. The blockchains are not maintained by any parties other than CAs, but by the bookkeepers which are operated also by CAs. So CertChain relies on reactive monitors to observe fraudulent certificates, which is similar to CT [52]. These schemes implement the transparency of key certification or certificate services in blockchains, but the key certification or certificate services are still controlled by entirely CAs as in traditional PKIs. On the contrary, our scheme integrates subject-controlled certificate publication with the transparencies of certificate signing and revocation, which actively tames the absolute authority of CAs.

Finally, our scheme is compatible with X.509 PKIs, while most of other blockchain-based certificate (or credential) management solutions do not consider the compatibility with legacy systems.

6 RELATED WORK

CT [52] uses append-only public log servers to ensure the accountability of CA operations. A certificate is accepted by browsers only if it is publicly recorded in the log servers, while monitors fetch all certificates from the logs and watch for suspicious ones. [22] formally defines the security properties of logging schemes, and proves that CT achieves these properties. B. Li *et al.* explored the TLS/HTTPS configurations of third-party monitors [54] and the reliability of third-party monitor services in the Internet [55].

RT [51], [73] extends the approach of CT to provide the public accountability of certificate revocation. CIRT [73] achieves both CT and RT, by recording certificates in two Merkle hash trees, one of which is in chronological order, and the other is lexicographic. Then, [77] improved CIRT by employing bilinear-map accumulators and binary trees, to achieve the same features but with shorter proofs. ARPKI [6] is proposed on top of CT. Multiple CAs and log servers work cooperatively to sign and publish certificates, while compromising any $n - 1$ out of n entities, does not result in successful attacks. CIRT, ARPKI and other publicly-accountable services [45], [51], [84] use Merkle hash trees to record certificates, and the append-only logs depend on extra

auditors. Our scheme is based on the append-only blockchain maintained by miners. CONIKS [61] presented transparent key directories based on Merkle prefix trees, allowing its users to audit their public keys while keep privacy. CoSi [83] forces certificates to be exposed and logged by a scalable group of witnesses, by a cosigning protocol. These schemes [61], [83] do not involve the subject control on publication so that a fraudulent certificate (or public key) might be accepted before it is observed, while our scheme supports subject-controlled certificate publication.

Subject-controlled certificate services were proposed, to balance the absolute authority of CAs. AKI [45] enables the certificate subject to define its own parameters (e.g., trusted CAs and log servers), and PoliCert [84] encodes these parameters as a subject certificate policy signed by multiple CAs. The subject certificate policy, can as well listed in Type-II transactions, to enhance the subject's control. DANE [39] enables a domain owner to specify the public keys of its trusted CAs and even its X.509 certificates, as DNSSEC-signed TLSA resource records [3], [5]. Similarly, its trusted CAs can be listed as CAA resource records [36], while a web server may enforce HTTPS access policies by DNSSEC HTTPSSR resource records [75] or HTTP STS header fields [38]. A sovereign key is registered by its owner (i.e., a web server) on timeline servers [26], and used to sign the web server's CA-signed certificate again. In these enhancements, a browser communicates with extra entities when validating a certificate, so privacy leakage happens (e.g., the visited server and the occurrence time); however, in our scheme, all browsers download uniform block-headers and no privacy information is exchanged.

Public key pinning requires browsers to locally store a public key (or certificate) for a certain domain [48]. Certificate Patrol [70] pins the certificate of an SSL/TLS website, and shows alerts if it receives different certificates. TACK [59] suggests a browser to pin another key, called the TACK signing key (TSK), if it observes the same TSK in two SSL/TLS sessions with a web server. The TSK is used by the SSL/TLS server to certify (or sign) the server's certificate again. HPKP [28] prompts browsers to pin public keys for the visited HTTP server. These schemes follow the assumption of trust on first use, so the first visit shall be established in the case of no attacks. Moreover, pinning-based mechanisms do not comprehensively consider revocation or update, while these events are handled as transactions in the certificate blockchain.

Notary-based approaches allow clients to compare certificates from different secure sessions, such as Perspectives [92], EFF SSL Observatory [27], and Convergence [58]. Crossbear [40] localises the SSL/TLS MitM attacks based on such records. A client may establish multiple SSL/TLS sessions over Tor [2], to detect fraudulent certificates by comparing the received certificates; or, clients exchange certificates received from different network paths [62]. Some notary-based approaches [27], [42], [58], [62], [92] leak privacy information about the secure sessions, while the others [2], [62] only work for localized attacks.

Browsers may enforce security-enhanced policies when validating certificates [1]. CA-TMS [8] evaluates the trustworthiness of CAs based on the client's local experiences, and then only a minimal set of CAs are trusted instead of the global settings of the Web PKI. CAGE [44] assigns a scope of

top-level domains to each CA. Any certificate signed by the CA with a domain name out of this scope, is considered as suspicious or invalid. Certlock [78] requires that, when a certificate is updated, the new certificate shall be signed by CAs from the same country as the previous one. DVCert [18] delivers a certificate list to browsers, protected by previously-established user credentials. Then, the list is used to validate certificates in the SSL/TLS negotiations. Such enhancements may be integrated into our scheme as additional rules to validate certificates.

Traditional certificate revocation approaches [16], [66] require extra communications by browsers, when they validate certificates in SSL/TLS negotiations. OCSP Stapling [24] enables browsers to receive OCSP responses as TLS extensions. Chrome CRLSet pushes CRLs as software updates [49], but it covers only a small portion of certificates [57]. RevCast [76] broadcasts revocation status through FM radio, while CRLite [50] aggregates revocation messages into a space-efficient data structure and pushes it to browsers. RITM [85] distributes revocation messages to the network middle-boxes which relay TLS handshakes and inserts them into the handshake packets. These approaches remove the extra communications for certificate revocation, and may be used to improve the coarse-grained RT in our scheme.

The SSL/TLS MitM attacks in the wild with forged certificates were investigated [41]. The surveys [13], [35] discussed the vulnerabilities of the SSL/TLS ecosystem and the countermeasures. [19] evaluated the practices of the certification guidelines by CA/Browser Forum, and [69] discussed the client certificate validation in SSL/TLS. Origin-bound certificates were proposed to strength client authentication [21]. [43] shows that, it is vulnerable to a new type of MitM attacks, and an improvement is proposed.

There are schemes to manage identities and public keys (or certificates) on top of blockchains. Namecoin [68] and its extensions [31], [32] allow the DNS domain owner to manage and certify its key pair by itself in the public blockchain, instead of some central authorities. Blockchain PKIs [53] and CertChain [12] record the CAs' operations of certification and revocation in two separate append-only blockchains, so that the transparencies of certification and revocation are achieved. Anyway, the above schemes still trust a single entity (i.e., the domain owner in [31], [32], [68] or the CA in [12], [53]) to control the relationship of the subject and its public key (or certificate), while our scheme distributes the trust on CAs in traditional PKIs to both the CAs and the community of web servers.

The Handshake project [87] integrates DANE and Namecoin, by securing the root zone file of DNSSEC in the PoW-based blockchain. It follows the spirit of Namecoin to distribute the trust among miners but only for the security of the root zone file (i.e., the top-level domains), while DANE is adopted for all subdomains. Therefore, the Handshake scheme works compatibly with existing lightweight DNSSEC clients. Handshake does not deal with the great number of non-top-level domains, while our scheme directly handles the certificates of each DNS name in a way compatible with X.509 PKIs.

Blockstack [65] improves Namecoin by separating controls and data in the blockchain-based systems. The control plane is built as a virtual blockchain on the Bitcoin blockchain, and

the Blockstack operations are encoded in Bitcoin transactions as additional metadata. This separation design can be integrated with our solution. Then, the community of web servers are only responsible for the operations of certificate transactions. An IKP reaction policy signed by issuers and a domain certificate policy signed by domains [60] construct a smart contract in the Ethereum blockchain. Then, a fraudulent certificate triggers the smart contract and involved parties receive payments. The game-theoretic analysis shows that, IKP incentivizes good CA behaviors and punishes misbehaviors.

There are various consensus protocols for blockchains, such as PoW [67], PoS [46], DPoS [86], and PBFT [9]. PoS and DPoS do not work for our scheme, because a cert-coin becomes invalid automatically after a certain period of time since it was generated, while PoS and DPoS encourage the participants to keep the crypto-currencies for a longer time. PBFT is unsuitable for Internet-scale blockchains. We adopt PoW in the proposed scheme, while the transaction delay of PoW does not matter in the certificate services where the status is generally not confirmed quickly.

Compared with the preliminary version [91], the mining incentives of the certificate blockchain is designed and analyzed in details, and we further analyze the attack scenario where CAs are compromised to sign fraudulent certificates and some certifiers are compromised to sign fraudulent Type II transactions.

7 CONCLUSION AND FUTURE WORK

We propose to record certificates and revocation status information in the global blockchain, which is inherently append-only, to achieve CT and coarse-grained RT. In the certificate blockchain, a certificate is periodically published as transactions by its subject, to balance the absolute authority of CAs. These transactions provide a continuous history of certificates for each SSL/TLS web server. The publishing key pairs used to sign certificate transactions, are also controlled cooperatively by CAs and the community of web servers, and recorded transparently in the blockchain. A series of regulations are designed to facilitate honest servers to sign countermeasure transactions and recover their certificates or publishing keys, under various attack scenarios. The scheme integrates CT, RT and subject-controlled certificate publication into blockchain-based append-only logs. It is compatible with X.509 PKIs but significantly reinforces the security guarantees of a certificate.

Our scheme provides transparent certificate validation delegate services for browsers. Before included in the certificate blockchain, each certificate chain published by the web server, is validated by a majority of P2P nodes (or honest servers). Therefore, an invalid certificate does not appear in the blockchain, and a browser only needs to check whether the root CA of the published certificate chain is trusted or not. The analysis based on the real-world statistics and the experimental results on the prototype system show that, our scheme introduces reasonable overheads, in terms of storage, certificate validation delay, communication, transaction verification delay, and incentive cost.

This paper presents the specification and the preliminary evaluation of this scheme, while the more formal security proofs will be included in our future work. The security

properties of transparent services (i.e., Bitcoin [67] and CT [52]) are formalized [11], [22], and we will follow the same spirits to further analyze the security of our scheme.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their detailed helpful reviews. This work was supported in part by Cyber Security Program (No. 2017YFB0802100) of National Key RD Plan of China, National Natural Science Foundation of China (No. 61772518), and National 973 Program of China (No. 2014CB340603). A preliminary version of this article appeared under the title "Blockchain-based Certificate Transparency and Revocation Transparency" in Proc. 5th Workshop on Bitcoin and Blockchain Research (BITCOIN), 2018 [91].

REFERENCES

- [1] M. Abadi, A. Birrell, I. Mironov, T. Wobber, and Y. Xie, "Global authentication in an untrustworthy world," in *Proc. 14th USENIX Conf. Hot Topics Operating Syst.*, 2013, Art. no. 19.
- [2] M. Alicherry and A. Keromytis, "DoubleCheck: Multi-path verification against man-in-the-middle attacks," in *Proc. 14th IEEE Symp. Comput. Commun.*, 2009, pp. 557–563.
- [3] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "IETF RFC 4033 - DNS security introduction and requirements," 2005.
- [4] C. Arthur, "Rogue web certificate could have been used to attack Iran dissidents," 2011. [Online]. Available: <https://iranian.com/main/news/2011/08/30/rogue-web-certificate-could-have-been-used-attack-iran-dissidents.html>
- [5] G. Ateniese and S. Mangard, "A new approach to DNS security (DNSSEC)," in *Proc. 8th ACM Conf. Comput. Commun. Secur.*, 2001, pp. 86–95.
- [6] D. Basin, C. Cremers, H. Kim, A. Perrig, R. Sasse, and P. Szalachowski, "ARPKI: Attack resilient public-key infrastructure," in *Proc. 21st ACM Conf. Comput. Commun. Secur.*, 2014, pp. 382–393.
- [7] bitcoin.org, Bitcoin developer guide, 2016.
- [8] J. Braun, F. Volk, J. Classen, J. Buchmann, and M. Mühlhäuser, "CA trust management for the Web PKI," *J. Comput. Secur.*, vol. 22, no. 6, pp. 913–959, 2014.
- [9] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proc. 3rd Symp. Operating Syst. Des. Implementation*, 1999, pp. 173–186.
- [10] Censys, "Censys public reports," 2016. [Online]. Available: <https://censys.io/>
- [11] M. Chase and S. Meiklejohn, "Transparency overlays and applications," in *Proc. 13th ACM Conf. Comput. Commun. Secur.*, 2016, pp. 168–179.
- [12] 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.*, 2018, pp. 2060–2068.
- [13] J. Clark and P. van Oorschot, "SoK: SSL and HTTPS: Revisiting past challenges and evaluating certificate trust model enhancements," in *Proc. 34th IEEE Symp. Secur. Privacy*, 2013, pp. 511–525.
- [14] Comodo CA Limited, "Comodo SSL certificates," 2017. [Online]. Available: <https://www.instantssl.com/ssl-certificate.html>
- [15] Comodo Group Inc. "Comodo report of incident," 2011. [Online]. Available: <https://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html>
- [16] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "IETF RFC 5280 - Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile," 2008.
- [17] N. T. Courtois and L. Bahack, "On subversive miner strategies and block withholding attack in Bitcoin digital currency," *ArXiv e-prints*, 2014.
- [18] I. Dacosta, M. Ahamad, and P. Traynor, "Trust no one else: Detecting MitM attacks against SSL/TLS without third-parties," in *Proc. 17th Eur. Symp. Res. Comput. Secur.*, 2012, pp. 199–216.
- [19] A. Delignat-Lavaud, M. Abadi, A. Birrell, I. Mironov, T. Wobber, and Y. Xie, "Web PKI: Closing the gap between guidelines and practices," in *Proc. 21st ISOC Netw. Distrib. Syst. Secur. Symp.*, 2014.

- [20] T. Dierks and E. Rescorla, "IETF RFC 5246 - The transport layer security (TLS) protocol," 2008.
- [21] M. Dietz, A. Czeskis, D. Balfanz, and D. Wallach, "Origin-bound certificates: A fresh approach to strong client authentication for the Web," in *Proc. 21st USENIX Secur. Symp.*, 2012, pp. 317–331.
- [22] B. Dowling, F. Gunther, U. Herath, and D. Stebila, "Secure logging schemes and certificate transparency," in *Proc. 21st Eur. Symp. Res. Comput. Secur.*, 2016, pp. 140–158.
- [23] R. Duncan, "Let's encrypt and Comodo issue thousands of certificates for phishing," 2017. [Online]. Available: <https://news.netcraft.com/archives/2017/04/12/lets-encrypt-and-comodo-issue-thousands-of-certificates-for-phishing.html>
- [24] D. Eastlake, "IETF RFC 6066 - Transport layer security (TLS) extensions: Extension definitions," 2011.
- [25] P. Eckersley, "A Syrian man-in-the-middle attack against Facebook," 2011. [Online]. Available: <https://www.eff.org/deeplinks/2011/05/syrian-man-in-the-middle-against-facebook>
- [26] P. Eckersley, "IETF Internet-draft - Sovereign key cryptography for Internet domains," 2012.
- [27] P. Eckersley and J. Burns, "Is the SSLiverse a safe place," 2010. [Online]. Available: <https://events.ccc.de/congress/2010/Fahrplan/events/4121.en.html>
- [28] C. Evans, C. Palmer, and R. Sleevi, "IETF RFC 7469 - Public key pinning extension for HTTP," 2015.
- [29] I. Eyal and E. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," in *Proc. 18th Int. Conf. Financial Cryptography Data Secur.*, 2014, pp. 436–454.
- [30] A. Freier, P. Karlton, and P. Kocher, "IETF RFC 6101 - The secure sockets layer (SSL) protocol version 3.0," 2011.
- [31] C. Fromknecht, D. Velicanu, and S. Yakoubov, "CertCoin: A NameCoin based decentralized authentication system," 2014. [Online]. Available: <http://courses.csail.mit.edu/6.857/2014/files/19-fromknecht-velicann-yakoubov-certcoin.pdf>
- [32] C. Fromknecht, D. Velicanu, and S. Yakoubov, "A decentralized public key infrastructure with identity retention," 2014. [Online]. Available: <https://eprint.iacr.org/2014/803.pdf>
- [33] M. Georgiev, S. Iyengar, S. Jana, R. Anubhai, D. Boneh, and V. Shmatikov, "The most dangerous code in the world: Validating SSL certificates in non-browser software," in *Proc. 19th ACM Conf. Comput. Commun. Secur.*, 2012, pp. 38–49.
- [34] GlobalSign, "Security incident report," 2011. [Online]. Available: <https://www.globalsign.com/resources/globalsign-security-incident-report.pdf>
- [35] A. Grant, "Search for trust: An analysis and comparison of CA system alternatives and enhancements," Dartmouth, Dept. Computer Science, Hanover, USA, Tech. Rep. TR2012-716, 2012.
- [36] P. Hallam-Baker and R. Stradling, "IETF RFC 6844 - DNS certification authority authorization (CAA) resource record," 2013.
- [37] E. Heilman, A. Kender, A. Zohar, and S. Goldberg, "Eclipse attacks on Bitcoin's peer-to-peer network," in *Proc. 24th USENIX Secur. Symp.*, 2015, pp. 129–144.
- [38] J. Hodges, C. Jackson, and A. Barth, "IETF RFC 6797 - HTTP strict transport security (HSTS)," 2012.
- [39] P. Hoffman and J. Schlyter, "IETF RFC 6698 - The DNS-based authentication of named entities (DANE) transport layer security (TLS) protocol: TLSA," 2012.
- [40] R. Holz, T. Riedmaier, N. Kammenhuber, and G. Carle, "X. 509 forensics: Detecting and localising the SSL/TLS men-in-the-middle," in *Proc. 17th Eur. Symp. Res. Comput. Secur.*, 2012, pp. 217–234.
- [41] L. Huang, A. Rice, E. Ellingsen, and C. Jackson, "Analyzing forged SSL certificates in the wild," in *Proc. 35th IEEE Symp. Secur. Privacy*, 2014, pp. 83–97.
- [42] ICSI, The ICSI certificate notary, 2011.
- [43] N. Karapanos and S. Capkun, "On the effective prevention of TLS man-in-the-middle attacks in web applications," in *Proc. 23rd USENIX Secur. Symp.*, 2014, pp. 671–686.
- [44] J. Kasten, E. Wustrow, and A. Halderman, "CAGE: Taming certificate authorities by inferring restricted scopes," in *Proc. 17th Financial Cryptography Data Secur. Conf.*, 2013, pp. 329–337.
- [45] T. Kim, L. Huang, A. Perrig, C. Jackson, and V. Gligor, "Accountable key infrastructure (AKI): A proposal for a public-key validation infrastructure," in *Proc. 22nd Int. Conf. World Wide Web*, 2013, pp. 679–690.
- [46] S. King and S. Nadal, "PPCoin: Peer-to-peer crypto-currency with proof-of-stake," 2012. [Online]. Available: <https://peercoin.net/assets/paper/peercoin-paper.pdf>
- [47] P. Koshy, D. Koshy, and P. McDaniel, "An analysis of anonymity in Bitcoin using P2P network traffic," in *Proc. 18th Int. Conf. Financial Cryptography Data Secur.*, 2014, pp. 469–485.
- [48] A. Langley, "Public key pinning," 2011. [Online]. Available: <https://www.imperialviolet.org/2011/05/04/pinning.html>
- [49] A. Langley, "Revocation checking and Chrome's CRL," 2012. [Online]. Available: <https://www.imperialviolet.org/2012/02/05/crlsets.html>
- [50] 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. 38th IEEE Symp. Secur. Privacy*, 2017, pp. 539–556.
- [51] B. Laurie and E. Kasper, "Revocation transparency," 2012. [Online]. Available: <http://sump2.links.org/files/RevocationTransparency.pdf>
- [52] B. Laurie, A. Langley, and E. Kasper, "IETF RFC 6962 - Certificate transparency," 2014.
- [53] K. Lewison and F. Coralla, "Backing rich credentials with a blockchain PKI," 2016. [Online]. Available: <http://pomcor.com/techreports/BlockchainPKI.pdf>
- [54] B. Li, D. Chu, J. Lin, Q. Cai, C. Wang, and L. Meng, "The weakest link of certificate transparency: Exploring the TLS/HTTPS configurations of third-party monitors," in *Proc. 18th IEEE Int. Conf. Trust Secur. Privacy Comput. Commun.*, 2019.
- [55] B. Li et al., "Certificate transparency in the wild: Exploring the reliability of monitors," in *Proc. 26th ACM Conf. Comput. Commun. Secur.*, 2019, pp. 2505–2520.
- [56] B. Li, J. Lin, Q. Wang, Z. Wang, and J. Jing, "Locally-centralized certificate validation and its application in desktop virtualization systems," *IEEE Trans. Inf. Forensics Security*.
- [57] Y. Liu et al., "An end-to-end measurement of certificate revocation in the Web's PKI," in *Proc. 15th Internet Meas. Conf.*, 2015, pp. 183–196.
- [58] M. Marlinspike, "Convergence," 2011. [Online]. Available: <https://github.com/moxie0/Convergence>
- [59] M. Marlinspike, "IETF Internet-draft - Trust assertions for certificate keys," 2013.
- [60] S. Matsumoto and R. Reischuk, "IKP: Turning a PKI around with decentralized automated incentives," in *Proc. 38th IEEE Symp. Security Privacy*, 2017, pp. 410–426.
- [61] M. Melara, A. Blankstein, J. Bonneau, E. Felten, and M. Freedman, "CONIKS: Bringing key transparency to end users," in *Proc. 24th USENIX Secur. Symp.*, 2015, pp. 383–398.
- [62] A. Micheloni, K. Fuchs, D. Herrmann, and H. Federrath, "Laribus: Privacy-preserving detection of fake SSL certificates with a social P2P notary network," in *Proc. 8th Int. Conf. Availability Rel. Secur.*, 2013, pp. 1–10.
- [63] Microsoft, "MS01-017: Erroneous VeriSign-issued digital certificates pose spoofing hazard," 2001. [Online]. Available: <https://technet.microsoft.com/library/security/ms01-017>
- [64] B. Morton, "Public announcements concerning the security advisory," 2013. [Online]. Available: <https://www.entrust.com/turktrust-unauthorized-ca-certificates>
- [65] A. Muneeb, N. Jude, S. Ryan, and J. Michael, "Blockstack: A global naming and storage system secured by blockchains," in *Proc. USENIX Annu. Tech. Conf.*, 2016, pp. 181–194.
- [66] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams, "IETF RFC 2560 - X. 509 Internet public key infrastructure online certificate status protocol - OCSP," 1999.
- [67] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: <https://bitcoin.org/bitcoin.pdf>
- [68] Namecoin Team, "Namecoin," 2011. [Online]. Available: <https://www.namecoin.org/>
- [69] A. Parsovs, "Practical issues with TLS client certificate authentication," in *Proc. 21st ISOC Netw. Distrib. Syst. Secur. Symp.*, 2014.
- [70] PSYC, "Certificate patrol," 2014. [Online]. Available: <http://patrol.psyced.org/>
- [71] M. Rosenfeld, "Analysis of Bitcoin pooled mining reward systems," *ArXiv e-prints*, 2011.
- [72] M. Rosenfeld, "Analysis of hashrate-based double spending," *ArXiv e-prints*, 2014.
- [73] M. Ryan, "Enhanced certificate transparency and end-to-end encrypted mail," in *Proc. 21st ISOC Netw. Distrib. Syst. Secur. Symp.*, 2014.
- [74] A. Sapirshstein, Y. Sompolinsky, and A. Zohar, "Optimal selfish mining strategies in Bitcoin," *ArXiv e-prints*, 2015.

- [75] S. Schechter, "IETF Internet-draft - Storing HTTP security requirements in the domain name system," 2007.
- [76] A. Schulman, D. Levin, and N. Spring, "RevCast: Fast, private certificate revocation over FM radio," in *Proc. 21st ACM Conf. Comput. Commun. Secur.*, 2014, pp. 799–810.
- [77] A. Singh, B. Sengupta, and S. Ruj, "Certificate transparency with enhancements and short proofs," in *Proc. 22nd Australas. Conf. Inf. Secur. Privacy*, 2017, pp. 381–389.
- [78] C. Soghoian and S. Stamm, "Certified lies: Detecting and defeating government interception attacks against SSL," in *Proc. 15th Financial Cryptogr. Data Secur. Conf.*, 2012, pp. 250–259.
- [79] Sosobtc, "Current Bitcoin and Litecoin price," May 2017. [Online]. Available: <https://www.sosobtc.com/?hl=en>
- [80] M. Spagnuolo, F. Maggi, and S. Zanero, "Bitlodine: Extracting Intelligence from the Bitcoin Network," in *Proc. 18th Int. Conf. Financial Cryptogr. Data Secur.*, 2014, pp. 457–468.
- [81] SSL Shopper, "SSL certificate for mozilla.com issued without validation," 2008. [Online]. Available: <https://www.sslshopper.com/article-ssl-certificate-for-mozilla.com-issued-without-validation.html>
- [82] SSL.com, "DV, OV, and EV certificates," 2015. [Online]. Available: <https://www.ssl.com/article/dv-ov-and-ev-certificates/>
- [83] E. Syta *et al.*, "Keeping authorities honest or bust with decentralized witness cosigning," in *Proc. 37th IEEE Symp. Security Privacy*, 2016, pp. 526–545.
- [84] P. Szalachowski, S. Matsumoto, and A. Perrig, "PoliCert: Secure and flexible TLS certificate management," in *Proc. 21st ACM Conf. Comput. Commun. Secur.*, 2014, pp. 406–417.
- [85] P. Szalachowski, L. Chuat, T. Lee, and A. Perrig, "RITM: Revocation in the middle," in *Proc. 36th Int. Conf. Distrib. Comput. Syst.*, 2016, pp. 189–200.
- [86] The BitShares Blockchain, "Delegated proof-of-stake consensus: A robust and flexible consensus protocol," 2018. [Online]. Available: <https://bitshares.org/technology/delegated-proof-of-stake-consensus>
- [87] The Handshake project, "Decentralized certificate authority and naming," 2019. [Online]. Available: <https://handshake.org/>
- [88] Unminer, "Current ETH price," May 2017. [Online]. Available: <http://www.unminer.com/tools/eth/calculator>
- [89] B. Vandersloot, J. Amann, M. Bernhard, Z. Durumeric, M. Bailey, and A. Halderman, "Towards a complete view of the certificate ecosystem," in *Proc. 16th Internet Meas. Conf.*, 2016, pp. 543–549.
- [90] VASCO Data Security International Inc., "DigiNotar reports security incident," 2011. [Online]. Available: https://www.vasco.com/about-vasco/press/2011/news_diginotar_reports_security_incident.html
- [91] Z. Wang, J. Lin, Q. Cai, Q. Wang, J. Jing, and D. Zha, "Blockchain-based certificate transparency and revocation transparency," in *Proc. 5th Workshop Bitcoin Blockchain Res.*, 2018, pp. 144–162.
- [92] D. Wendlandt, D. Andersen, and A. Perrig, "Perspectives: Improving SSH-style host authentication with multi-path probing," in *Proc. USENIX Annu. Tech. Conf.*, 2008, pp. 321–334.
- [93] K. Wilson, "Distrusting new CNIC certificates," 2015. [Online]. Available: <https://blog.mozilla.org/security/2015/04/02/distrusting-new-cnnic-certificates/>
- [94] L. Zhang *et al.*, "Analysis of SSL certificate reissues and revocations in the wake of Heartbleed," in *Proc. 14th Internet Meas. Conf.*, 2014, pp. 489–502.
- [95] M. Zusman, "Criminal charges are not pursued: Hacking PKI," 2009. [Online]. Available: https://www.defcon.org/images/defcon-17/dc-17-presentations/defcon-17-zusman-hacking_pki.pdf

Ze Wang received the PhD degree from the Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China. He works with Huawei Technologies company, Ltd. His research interests include trust management and public key infrastructures.

Jingqiang Lin (Senior Member, IEEE) received the MS and PhD degrees from the Graduate University of Chinese Academy of Sciences, Beijing, China, in 2004 and 2009, respectively. He is currently a full professor with the Institute of Information Engineering, Chinese Academy of Sciences. His research interests include system security and applied cryptography.

Quanwei Cai is currently an assistant professor with the Institute of Information Engineering, Chinese Academy of Sciences. His research interests include trust management and public key infrastructures.

Qiong Xiao Wang is currently an associate professor with the Institute of Information Engineering, Chinese Academy of Sciences. Her research interests include trust management and public key infrastructures.

Daren Zha is currently an associate professor with the Institute of Information Engineering, Chinese Academy of Sciences. His research interests include trust management and public key infrastructures.

Ji Wu Jing (Member, IEEE) received the MS and PhD degrees from the Graduate University of Chinese Academy of Sciences, Beijing, China. He is currently a full professor with the School of Computer Science and Technology, University of Chinese Academy of Sciences. His research interests include network and system security.

► **For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.**