Blockchain-based Certificate Transparency and Revocation Transparency*

Ze Wang[1,2,3], Jingqiang Lin[1,2,3**], Quanwei Cai[1,2], Qiongxiao Wang[1,2,3], Jiwu Jing[1,2,3], and Daren Zha[1,2]

[1] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China.
[2] Data Assurance and Communication Security Research Center, Chinese Academy of Sciences, Beijing 100093, China.
[3] School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China.
{wangze,linjingqiang,caiquanwei,wangqiongxiao}@iie.ac.cn

Abstract. Traditional X.509 public key infrastructures (PKIs) depend on 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 paper, we propose blockchain-based certificate transparency and revocation transparency. 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, and miners of the community append it to the global certificate blockchain after verifying the transaction and mining a block. 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 to balance the absolute authority of CAs in traditional PKIs. We implement the prototype system with Firefox and Nginx, and the experimental results show that it introduces reasonable overheads.

Keywords: PKI, SSL, TLS, blockchain, transparency, trust

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 (typically a DNS name). Then, these certificates are used in SSL/TLS [21, 14] to authenticate the

* This work was partially supported by National Basic Research 973 Program of China (Award No. 2014CB340603), National Natural Science Foundation of China (Award No. 61772518), and Cyber Security Program (Award No. 2017YFB0802100) of National Key RD Plan of China.

** Corresponding author.

web servers. Trusting the CAs, browsers obtain the servers' public keys from CA-signed certificates in SSL/TLS negotiations, to establish secure channels.

However, recent security incidents indicate that CAs are not so trustworthy as they are assumed to be. CAs may sign fraudulent certificates due to intrusions [11, 24, 58, 46], reckless identity validations [44, 61, 54, 55], misoperations [45, 62, 33], flawed cryptographic algorithms [53, 60], or government compulsions [15, 52]. Typical fraudulent certificates bind a DNS name to key pairs held by counterfeit web servers [5, 11, 15]. Then, the counterfeit servers will launch man-in-the-middle (MITM) attacks, even when a browser follows the strict steps of certificate validation [12] to establish SSL/TLS sessions.

Certificate transparency [35] was proposed to enhance the accountability of CA operations, using *append-only* public logs. A CA-signed certificate is publicly recorded in log servers; otherwise, a browser rejects it in SSL/TLS negotiations. Therefore, a fraudulent certificate will be observed by monitors or interested parties, especially the owner of the DNS name (or the SSL/TLS web server). The certificates in log servers are organized as a Merkle hash tree, and auditors periodically verify the integrity of logs to ensure that they are append-only, i.e., a (fraudulent) certificate will never be deleted or modified after being appended.

Certificate transparency follows a reactive philosophy. It depends on the monitors to observe fraudulent certificates after they have been signed and recorded in the log servers. So a fraudulent certificate may be accepted by browsers before it is observed by any interested party. Moreover, to ensure append-only records, and the detection of deleted or modified (fraudulent) certificate relies on the periodical detection of auditors. Considering the huge number and increment of current certificates, it is a high burden to monitor or audit the public logs.

In this paper, we proposes a blockchain-based scheme to construct append-only logs for certificate transparency. In our scheme, CA-signed X.509 certificates are published by their subjects (i.e., the corresponding web servers) in *certificate transactions* in a *global certificate blockchain*. The global certificate blockchain acts as an inherently append-only public log to monitor CAs' operations. To publish its certificates, each web server has a *publishing key pair* to sign its certificate transactions, which is different from the key pair bound in the certificate. This design of subject-controlled certificate publication allows an web server to manage its certificates cooperatively with CAs. It shares the same spirit with trust assertions for certificate keys (TACK) [39] and PoliCert [56] that a web server is involved in the validation of its certificates.

SSL/TLS web servers compose an interdependent community, to balance the absolute authority of CAs in traditional PKIs. Particularly, the publishing key of a web server is initially certified by a certain number of these interdependent web servers explicitly and also a CA implicitly. So CAs are unable to publish a certificate in the certificate blockchain, without the consent of the community of web servers. The certified publishing key of each web server is also publicly recorded in the blockchain. It means that the publication of a certificate is also publicly accountable.

Each certificate transaction has a period of validity, shorter than those of the published certificates. It enforces a web server to publish its certificates and update revocation status periodically. When an unexpired certificate is revoked, it will be absent from the next transaction, and the corresponding certificate revocation list (CRL) file or online certificate status protocol (OCSP) response will be included in the transaction instead. Therefore CAs' revocation operations are also recorded and revocation transparency [34, 51] is achieved in the certificate blockchain.

The proposed scheme protects browsers against the impersonation attacks using fraudulent certificates. Browsers utilize the certificate blockchain to validate the certificates received in SSL/TLS negotiations. A certificate is accepted only if it is published in an unexpired transaction, therefore a fraudulent certificate signed by compromised CAs but not published in the blockchain will be rejected by browsers.

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, and incentive cost.

The remainder is organized as follows. In Section 2, we introduce the security model, 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 is 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. We assume that the attackers could compromise some trusted CAs, to sign fraudulent certificates binding the target web server's DNS name to any key pair. The attackers may also compromise the target server's publishing key to generate fraudulent messages. However, we assume that they could not compromise more than a threshold of certifiers of the target server and certify a fraudulent publishing key. That is, the worst scenario our scheme considers is that the publishing key pair is held or known by attackers and fraudulent certificates are signed by compromised CAs at the same time.

The attackers do not hold computation resource which exceeds 30% of all computation power in the community, and all cryptographic primitives are secure. The attackers cannot block the network for a long enough time to take attack actions; that is, honest entities communicate with each other in a loosely synchronized manner.

We aims to protect browsers against the impersonation attacks using fraudulent certificates. If a browser follows our scheme to validate 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 browsers are most likely to reject this fraudulent certificate and it will be recovered soon by countermeasure transactions in the certificate blockchain.

3 System Architecture

3.1 Overview

Web servers publish their certificate transactions in an unique, global certificate blockchain. Like Bitcoin, each block in the certificate blockchain consists of a block header and multiple transactions, which are organized as a Merkle hash tree. 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 in the blockchain, *c*) the Merkle hash tree root of the included transactions, *d*) a list of (`Type`, `DNS_Name`) tuples of the included transactions, sorted in lexicographic order, and *e*) a PoW nonce computed by brute force, so that the hash value of this block is less than the PoW target. The list of (`Type`, `DNS_Name`) tuples is included, so browsers or web servers find a transaction conveniently without iterating through all included transactions.

In the certificate blockchain, there are overall 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; at the same time, the corresponding CRL file or OCSP 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 other web servers (called *certifiers*) using their publishing key pairs. This publishing key may be reset by another Type-II transaction, if it is compromised or lost.

All certificate transactions labeled with the same DNS name, either Type-I or Type-II, are chained together chronologically, as shown in Figure 1. For every web server, its continuous history of certificates and publishing keys is then archived publicly in the blockchain. If a fraudulent certificate or publishing key appears, it will be observed soon. We design a series of regulations that enable the honest web servers to take countermeasures and recover their certificates or publishing keys.

A miner collects, verifies certificate transactions from others and mines the block for them, according to the regulations of certificate transaction stated as below. It also collects mined blocks from other miners, verifies the blocks, and appends valid ones to its local copy to make it be longer. Incentive mechanism like Bitcoin can be introduced to our scheme to encourage mining. Namely, the miner who mines a particular block will be awarded by some coins, and the coins are purchased and consumed by the web servers when they publish certificate transactions.

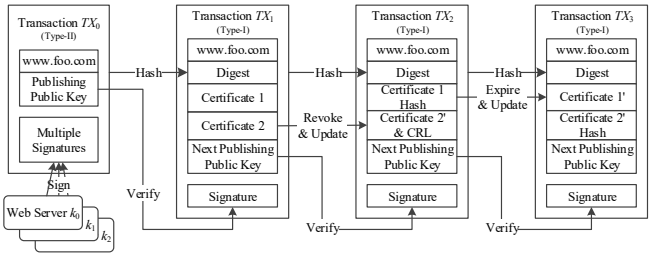

Fig. 1. The history of certificates and publishing keys.

In every block, a web server is allowed to publish at most one transaction. If the miners of the blockchain receive both a Type-I and a Type-II transaction 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 implementation.

A browser communicates with the P2P network of web servers/miners to download the updated blockchain, to keep a local copy. When the browser is establishing SSL/TLS sessions with a web server, it validates the web server's certificate with the help of its local copy of the certificate blockchain. We will explain this procedure in further details in the following subsections.

3.2 Subject-controlled Certificate Publication

A certificate is signed and published by its subject, in Type-I certificate transactions. Only the web sever who holds the publishing key pair, is able to sign Type-I transactions labelled with its DNS name. Therefore, a (compromised) CA cannot publish certificates without the consent of the subject. 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, currently valid certificates.
6. `Next_Publishing_Key`, the publishing public key. The corresponding private key is used to sign the *next* Type-I transaction with this 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).

For each certificate chain in `List_of_Cert_Chain`, if the web server's certificate is published for the first time, the whole chain from the self-signed root

CA certificate to the end-entity certificate is provided. It facilitates the miners to validate the certificate. Otherwise, when the chain is published in the subsequent periodical Type-I transactions, only a simplified entry is provided, including the hash value of the web server's certificate and the information to check its "dynamic" validity status. Such information includes the validity of certificate, the CRL distribution point or OCSP location, and the identifier to check its revocation status via CRL or OCSP (e.g., the serial number). Such design of simplified entries enables the miners to verify the consistency of these transactions and check the certificate's dynamic revocation status, but greatly reduces the overheads of storage and communication.

Certificate transparency is achieved since certificates are publicly visible as transactions in the append-only blockchain. For Type-I transactions, their period of validity is required to be comparable with that of certificate status refresh in current PKI. Therefore, a certificate is usually published for several times during its lifecycle, and its updated revocation status is also reflected in certificate transactions. When a certificate is revoked, the web server replaces this certificate with related revocation information in the next transaction, and publish it immediately or until the next period. So CA's revocation operations are also recorded in the blockchain, which enables *revocation transparency*.

3.3 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. 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 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 remove fraudulent certificates falsely appeared in previous Type-I transactions in emergency.
7. `Sig_by_Owner`, some signatures by the certified web server. They are verified using corresponding some CA-signed certificates binding the DNS name, which are also included in this field.
8. `List_of_Sig`, a list of signatures signed by certifier web servers using their current publishing key pairs. The signers' DNS names are also in this field.

Each Type-II transaction is signed by at least G certifier web servers. 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 it trusted whose publishing keys

have been certified in the blockchain, as its certifiers. The *initial* Type-II transaction is signed by *all* certifiers. Then, any following *non-initial* Type-II transaction must be signed by at least G certifiers specified in `Certifier_Group` of the previous Type-II transaction. The certifier group can be modified in *non-initial* Type-II transactions, while the number of the certifiers in `Certifier_Group` must be not less than G. Besides, Type-II transactions are also signed by the web server itself in `Sig_by_Owner`, which indirectly certified by one or more CAs.

An initial Type-II transaction is set with a "frozen" period before the certified publishing key becomes valid. It allows interested parties (not only the web server) to observe impersonation attack attempts. During the frozen period, another initial Type-II transaction with the same DNS name whose certificates in `Sig_by_Owner` are signed by more publicly-trusted CAs invalidates this Type-II transaction. If such dispute case happens, the frozen period is automatically extended to provide the target web server more time to contact CAs and out-of-band actions may be taken. The publishing key becomes valid, after the frozen period without disputes.

The field of `List_of_Cert_Chain` is used in the extreme attack case that fraudulent certificates are published by a compromised publishing key pair, so the target web server recovers its control on publishing keys and delete fraudulent certificates by only one transaction. No revocation information is needed in this field of Type-II transactions. Note that, `List_of_Cert_Chain` is not allowed to include any newly appeared certificate; otherwise, it provides a fast track for attackers to publish fraudulent certificates.

To reduce the storage requirement of miners, each web server publishes its Type-II transactions *periodically*, even when it is unnecessary to reset the publishing key pair or modify its certifier group. In these "shadow" Type-II transactions, `Publishing_Key` and `Certifier_Group` must be identical with those in the previous transaction, while `List_of_Cert_Chain` and the certificates in `Sig_of_Owner` must be absent. A shadow Type-II transaction is signed only by the web server itself, and no signatures by certifiers are required. So, a miner only stores *a*) the recent block headers and *b*) the latest transactions of both types of each DNS name, within a certain period.

In the *genesis* block of the certificate blockchain, $G+1$ or more special Type-II transactions are included to certify at least $G+1$ web servers' publishing keys. Each of these transactions, is signed by other G web servers. The publishing keys in the genesis block are self-certified by these web servers cooperatively (and some CAs indirectly). Then, these web servers will publish their certificates.

An initial period shall be needed for the founder servers to invite highly-ranked web servers (assumed to be honest) to join, i.e., initially certify other web servers' publishing keys. During the initial period, the certificate blockchain is publicly visible but the community is not open to join. After a certain number of honest web servers are introduced into the community, so that any new participants will not introduces overwhelmed computation power, it is open to the public and the non-founder web servers as well operate as certifiers after their publishing keys have been certified and published in the certificate blockchain.

3.4 Certificate Validation by Browsers

A browser validates the certificates received in SSL/TLS negotiations, with the help of the certificate blockchain. First, a browser communicates with (the P2P network of) web servers/miners to incrementally download the up-to-date block headers. 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. Browsers download and store only block headers but no transactions, to reduce the overhead of communication and storage. 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 via SSL/TLS message extensions. 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 [35]) are also sent to enable browsers to verify the certificate transaction.

A certificate chain received in SSL/TLS negotiations is valid, if *a*) the certificate (or its hash value) is published in `List_of_Cert_Chain` of an *unexpired* transaction, which is 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, not in the latest N ones of the blockchain, to ensure that the certificate transaction has been accepted by enough miners [8] 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 are any transactions with the visited DNS name, then the received certificate shall also appear in these transactions. Note that browsers are immune to downgrading attacks, since they will determine whether a web server has published transaction in the blockchain according to their local copies of block headers, and perform standard certificate validation for those servers who has not.

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 [12] such as CA signatures, periods of validity, and certificate extensions, is unnecessary. That is, the certificate validation of browsers is delegated to the community of miners, and the delegated certificate validation is also transparent (i.e, publicly visible).

Since the validity period of Type-I transactions may be greater than the general CRL/OCSP update period, a browser with high security concerns may take extra actions to check the revocation status of the received certificates.

4 Security Analysis

In this section, we analyze the proposed scheme under various attack scenarios. We present the countermeasures and evaluate the impacts, when some entities except the target web server were compromised. The network attacks on the certificate blockchain are also analyzed.

4.1 Security with Compromised Key Pairs

In order to impersonate a web server, the attackers could compromise a CA, and/or compromise some web servers' publishing key pairs. Note that, an attack is considered as successful only if a fraudulent certificate is accepted by browsers.

We do not assume that the attackers could compromise at least G certifiers of the target web server, considering the certifiers are carefully selected by the target server, and G can be set large enough. We neither consider the situation that the key pair bound in the target web server's certificate is compromised by attackers, in which the attackers could launch MITM attacks without a fraudulent certificate. Such attacks should be prevented or mitigated by the approaches other than certificate management, which are out of the scope of this paper.

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 in the blockchain. Note that in traditional PKIs compromising a CA is sufficient to launch MITM attacks, while in our scheme attackers need to concurrently compromise the target server's publishing key, which is more difficult.

Compromised Publishing Key Pairs. An attacker fails to impersonate the target web server, if it only compromises its publishing key pair. The attacker may modify `List_of_Cert_Chain` in a Type-I transaction, by including an expired or revoked one, or excluding a currently-valid certificate. But the verification of such a transaction will fail, because miners will validate all included certificates and only expired or revoked certificates shall be excluded compared with the previous transaction.

The attacker may modify `Next_Publishing_Key` with another key pair through Type-I transactions, to prevent the web server from updating its Type-I transactions. This case will be further discussed in the following paragraphs.

Compromised CAs and Publishing Key Pairs. The attackers might compromise a CA to sign fraudulent certificates and also compromise the target web server's publishing key pair. Then, the attacker could publish fraudulent certificates in a Type-I transaction, while simultaneously modify `Next_Publishing_Key` with another key pair held by itself. This transaction will be considered as correct by miners, and finally included into a mined block.

Once the fraudulent Type-I transaction is observed by the target web server (or any other interested parties), the web server will immediately contact its certifiers to sign a countermeasure Type-II transaction, to exclude the fraudulent certificates by properly setting `List_of_Cert_Chain` and to simultaneously reset its publishing key in `Publishing_Key`. As shown in Figure 2, if the countermeasure Type-II 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 3.4). 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.

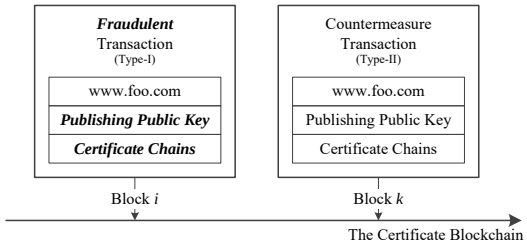

Fig. 2. A fraudulent Type-I transaction in Block i, and the countermeasure Type-II transaction in Block k. If $i < k \leq i + N$, the fraudulent certificate is never accepted; if $k > i + N$, it may be accepted before the countermeasure transaction.

In all above scenarios, there is no attack impact if target web servers observe attacks and take countermeasures in time (i.e. $k \leq i + N$). Since attackers can not compromise G certifiers, a web server can always counters the impact by a Type-II transaction.

4.2 Attacks on the Certificate Blockchain

The attackers may attempt to prevent browsers from accessing the recent blocks of the certificate blockchain. First, it is extremely difficult and expensive to isolate a browser from a great number of P2P nodes, while allow it to access (counterfeit) web servers. Even if such an attack could be performed, the victim browser is aware of it when subsequent blocks take far more time than average block interval, since the mining time is contained in block headers.

Powerful attackers might publish a fraudulent certificate and include it in a branch of the blockchain. Then, subsequent blocks are mined on the branch by attackers, while countermeasure transactions are elaborately excluded. It requires the attackers to control 33% of computation resources [20], which breaks our security assumption on attackers' computation power [48].

5 Implementation and Evaluation

This section presents the prototype of the proposed scheme, and evaluates its performance.

5.1 Implementation and Setting

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

The browser is implemented on Firefox Nightly (version 54.0a1). The function `VerifySSLServerCert()` is modified as described in Section 3.4 to validate the

received certificate. The web server is implemented on Nginx (version 1.10.3). Extensions of ClientHello and ServerHello are defined and implemented in the handshake of SSL/TLS, to request and respond the certificate transactions and the corresponding Merkle audit paths. The browser and the visited web server are connected directly. The browser runs on a desktop (Intel i7-4770s/3.10GHz CPU, 8GB RAM, and ST-1000DM003 hard disk) with Windows 7 Professional, and the web server is on another desktop of the same hardware configuration with Ubuntu 16.04 LTS (32-bit).

Table 1 summarizes the values of all related parameters in the prototype. The reasons why we choose these values are explained in the appendix in detail.

Table 1. The values of parameters in the prototype blockchain.

Mark	**Value**	**Note**
T_B	2 hours	Average block interval.
T_I	10 days	Type-I transaction validity.
T_{II}	100 days	The period of shadow Type-II transactions.
G	5/10	The threshold of signed certifier in a Type-II transaction.
N	6	Number of blocks after a block to fully confirm the block.

There are currently about 54.35M valid SSL/TLS certificates according to Censys.io [43], and on average each website has 1.34 certificates [57]. Nearly 70% of these certificates are issued for free (e.g. by Let's Encrypt). We think these certificates' owners are sensitive to price, and would not participate in the incentive mechanism in our scheme. Thus, we mainly focus on the non-free certificates (approximately 13.88M, corresponding to about 10.36M websites). Note that we do not require all certificates to participate since our scheme is immune to downgrading attack. We include all these certificates basically for performance estimation.

We generate a prototype blockchain using 10.36M random DNS names, and average-size certificates by testing CAs. The prototype blockchain contains 1200 blocks (within about T_{II}) and transactions for all 10.36M web servers. In the prototype, each server periodically publishes a 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. On average each web site has 1.34 certificate chains and each chain is composed of three certificates. We use OpenSSL-1.0.1g to generate the prototype certificate blockchain. Besides, DNS names are randomly generated in the prototype blockchain with the average length of 14 bytes, according to the average domain name length of Alexa top 1M websites [2].

OCSP responses (1.60KB on average) are included as the revocation status information, because of OCSP is widely deployed [37] and smaller than CRL (51KB on average [37]). According to the real-world statistics, the revocation rate is normally 1% and may increase to 11% when an emergency happens [37].

5.2 Evaluation

Storage. Table 2 lists the average size of each element. A browser keeps the block headers within T_I for certificate validation, so the storage overhead is about $120 \times 1.40\text{MB} \approx 168.00\text{MB}$.

Table 2. The size of data elements in the blockchain.

Item	Average Size
Type-I transaction	
with 1/2/3 certificate chain(s)	4.65/8.71/12.77KB
with 1/2/3 partial certificate(s)	0.83/1.07/1.30KB
Type-II certificate transaction	
without certifier signature (shadow)	1.02KB
with 5/10 certifier's signatures	5.10/5.36KB
Block header	559.18KB
Block, including the header and transactions	
no certificate revoked	95.05MB
1% certificates revoked (by OCSP/CRL) and 1% new certificates	101.05/162.62MB
11% certificates revoked (by OCSP/CRL) and 11% new certificates	161.13/838.42MB

A miner keeps the latest Type-I and Type-II transactions of each DNS name, and all block headers within T_{II}, which is about $10.36\text{M} \times (0.91\text{KB} + 1.02\text{KB}) + 1200 \times 4 \times 1.40\text{MB} \approx 26.64\text{GB}$. Here, 0.91KB is the estimated size of Type-I transaction with 1.34 certificate chains. Besides, 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 about $6 \times (93.06\text{K} \times 0.91\text{KB} + 8.84\text{K} \times 1.02\text{KB}) \approx 0.55\text{GB}$. That is, a miner stores at least 27.19GB data. If 1% certificates are revoked by OCSP, the storage overhead increases to 27.36GB (we consider it as a "typical" size); if 11% are revoked by CRL, the overhead is 109.62GB. Even when all valid certificates are included in the blockchain, the typical storage overhead of a miner will not exceed 150GB. As a comparison, each miner in Bitcoin stores over 150GB data.

Delay. A browser validates the received certificate 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 blockchain (each for 1000 times). The average time of these operations is listed in Table 3.

The time cost is greatly reduced if a browser loads all block headers (about 168.00MB) into memory when it starts up. In such case, Operation *b*) is un-

Table 3. The time of certificate validation in browsers.

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

necessary and the average delay is 0.98ms. This number will increase to 3.20ms when all valid certificates are included in the blockchain.

The browser takes about 9.86ms to validate a certificate using the standard method except checking revocation. Adopting our method may significantly accelerate the validation process.

Communication. In an SSL/TLS negotiation, the introduced communication overhead is typically one certificate transaction, and will not exceeds $N + 1$ transactions. It is about 1.53KB, and not larger than 10.71KB ($N = 6$).

Besides, a browser periodically updates its local copy of block headers, about 16.80MB every day. The number can be decreased to about 5∼8MB daily via common commercial compress software. Compared to traditional methods who need extra links to download CRLs/OCSPs during SSL/TLS negotiations, which degrades performance even they are small, our scheme enable browsers to download block headers when they are idle.

6 Related Work

Several **public-accountable-log-based services**, such as CIRT [51], RT [34], ARPKI [7], and CONIKS[41], have been introduced to achieved certificate transparency and/or revocation transparency, by recording certificates in Merkle hash trees. These schemes do not involve subject control on certificate publication so that a fraudulent certificate (or public key) might be accepted before it is observed by interested parties, while our blockchain-based scheme supports subject-controlled certificate publication. AKI [31] and Policert [56] enable the certificate subject to define its own certificate parameters (e.g., trusted CAs and log servers). The subject certificate policy, can as well listed in Type-II transactions, to enhance the subject's control. All above append-only logs depend on extra auditors which are not needed in our scheme. On the contrary, our scheme is based on the append-only blockchain maintained by web servers and miners.

Other **subject-controlled certificate services**, like DANE [27], CAA [26], Sovereign Key [16] were also proposed to balance the absolute authority of CAs. In these enhancements, a browser communicates with extra components

(a DNSSEC server [6, 4] or a timestamp server) when validating a certificate or public key, so privacy information about the secure session is leaked; however, in our scheme, all browsers download uniform block headers and no such information is exchanged.

Public key pinning requires browsers to locally store a public key (or certificate) for a certain domain [32, 50, 19, 39]. These schemes follow the assumption of trust on first use, so the first visit shall be established without attacks. Moreover, existing pinning mechanisms do not consider certificate revocation or update, while these events are handled as transactions in our scheme.

Notary-based approaches allow clients to compare certificates from the SSL/TLS sessions and other sources, such as Perspectives [59], Convergence [38], ICSI Notary[29], and EFF SSL Observatory [17]. Crossbear [28] localises the SSL/TLS MITM attacks based on such records. Doublecheck [3, 18] and Laribus [42] allow clients compares certificates received from different network paths. Some notary-based approaches [59, 38, 29, 17, 42] leak the privacy information about secure sessions, while the others [3, 18, 42] only work for localized attacks.

Browsers may enforce **enhanced security policies** when validating a certificate [1]. CA-TMS [9], Certlock [52] and Cage [30] separately evaluate the trustworthiness of CAs based on the client's local experiences or the CA's domain name scope. DVCert [13] delivers a certificate list to browsers, protected by previously-established user credentials, to validate certificates in the SSL/TLS negotiations. Such enhancements may be integrated into our scheme as additional rules to validate certificates. The surveys [25, 10] comprehensively discussed the vulnerabilities of the SSL/TLS ecosystem and the countermeasures.

Some **blockchain-based alternatives** allow subjects to publish keys or credentials in blockchain [49, 22, 23, 36]. In these DNS systems [49, 22, 23], the key pairs are controlled entirely by the owner of the DNS name. So the private key can not be recovered ever since it is compromised. Our solution distributes the control among CAs and the community of web servers, so an attacker cannot bind an arbitrary key pair to the DNS name after breaking a single entity. An IKP reaction policy signed by issuers and a domain certificate policy signed by domains construct a smart contract in the Ethereum blockchain [40], intending to be triggered by fraudulent certificates. Blockstack [47] improves Namecoin by separating controls and data. This separation design can be integrated with our solution, making the storage of certificate transactions outsourced.

7 Conclusion

We propose to record certificates and revocation status information in the global certificate blockchain, which is inherently append-only, to achieve certificate transparency and limited-grained revocation transparency. Our scheme balances the absolute authority of CAs, and provides a continuous history of certificates for each SSL/TLS web server. The publishing key pairs used to sign transactions, are controlled cooperatively by CAs and the community of web servers, and recorded in the blockchain. The proposed scheme is compatible with X.509

PKIs but significantly reinforces the security guarantees of certificates. Our scheme also provides transparent and delegated certificate validation services for browsers. Since each certificate chain is validated by a majority of miners before included in the certificate blockchain. The analysis and the experimental results show that, our scheme introduces reasonable overheads in terms of storage, certificate validation delay, communication, and incentive cost.

References

1. Abadi, M., Birrell, A., Mironov, I., Wobber, T., Xie, Y.: Global authentication in an untrustworthy world. In: 14th USENIX Conference on Hot Topics in Operating Systems (HotOS) (2013)
2. Alexa: Alexa top 1M websites (2017), http://s3.amazonaws.com/alexa-static/top-1m.csv.zip
3. Alicherry, M., Keromytis, A.: Doublecheck: Multi-path verification against man-in-the-middle attacks. In: 14th IEEE Symposium on Computers and Communications (ISCC). pp. 557–563 (2009)
4. Arends, R., Austein, R., Larson, M., Massey, D., Rose, S.: DNS security introduction and requirements. Tech. rep., IETF RFC 4033 (2005)
5. Arthur, C.: Rogue web certificate could have been used to attack Iran dissidents (August 2011), https://iranian.com/main/news/2011/08/30/rogue-web-certificate-could-have-been-used-attack-iran-dissidents.html
6. Ateniese, G., Mangard, S.: A new approach to DNS security (DNSSEC). In: 8th ACM Conference on Computer and Communications Security (CCS). pp. 86–95 (2001)
7. Basin, D., Cremers, C., Kim, H., Perrig, A., Sasse, R., Szalachowski, P.: ARPKI: Attack resilient public-key infrastructure. In: 21st ACM Conference on Computer and Communications Security (CCS). pp. 382–393 (2014)
8. bitcoin.org: Bitcoin developer guide (2016), https://bitcoin.org/en/developer-guide
9. Braun, J., Volk, F., Classen, J., Buchmann, J., Mühlhäuser, M.: CA trust management for the web PKI. Journal of Computer Security 22(6), 913–959 (2014)
10. Clark, J., van Oorschot, P.: SoK: SSL and HTTPS: Revisiting past challenges and evaluating certificate trust model enhancements. In: 34th IEEE Symposium on Security and Privacy (S&P). pp. 511–525 (2013)
11. Comodo Group Inc.: Comodo report of incident (March 2011), https://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html
12. Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., Polk, W.: Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile. Tech. rep., IETF RFC 5280 (2008)
13. Dacosta, I., Ahamad, M., Traynor, P.: Trust no one else: Detecting MITM attacks against SSL/TLS without third-parties. In: 17th European Symposium on Research in Computer Security (ESORICS). pp. 199–216 (2012)
14. Dierks, T., Rescorla, E.: The transport layer security (TLS) protocol. Tech. rep., IETF RFC 5246 (2008)
15. Eckersley, P.: A syrian man-in-the-middle attack against facebook (May 2011), https://www.eff.org/deeplinks/2011/05/syrian-man-middle-against-facebook

16. Eckersley, P.: Sovereign key cryptography for internet domains. Tech. rep., IETF Internet-draft (2012)
17. Eckersley, P., Burns, J.: Is the SSLiverse a safe place (December 2010), https://events.ccc.de/congress/2010/Fahrplan/events/4121.en.html
18. Engert, K.: Detector (September 2013), http://www.detector.io/DetecTor.pdf
19. Evans, C., Palmer, C., Sleevi, R.: Public key pinning extension for http. Tech. rep., IETF RFC 7469 (2015)
20. Eyal, I., Sirer, E.G.: Majority is not enough: Bitcoin mining is vulnerable pp. 436–454 (2013)
21. Freier, A., Karlton, P., Kocher, P.: The secure sockets layer (SSL) protocol version 3.0 (2011)
22. Fromknecht, C., Velicanu, D., Yakoubov, S.: Certcoin: A namecoin based decentralized authentication system (2014), http://courses.csail.mit.edu/6.857/2014/files/19-fromknecht-velicann-yakoubov-certcoin.pdf, massachusetts Institute of Technology, MA, USA
23. Fromknecht, C., Velicanu, D., Yakoubov, S.: A decentralized public key infrastructure with identity retention (2014), https://eprint.iacr.org/2014/803.pdf
24. GlobalSign: Security incident report (2011), https://www.globalsign.com/resources/globalsign-security-incident-report.pdf
25. Grant, A.: Search for trust: An analysis and comparison of CA system alternatives and enhancements. Tech. rep., Dartmouth Computer Science, Technical Report TR2012-716 (2012)
26. Hallam-Baker, P., Stradling, R.: DNS certification authority authorization (CAA) resource record. Tech. rep., IETF RFC 6844 (2013)
27. Hoffman, P., Schlyter, J.: The DNS-based authentication of named entities (DANE) transport layer security (TLS) protocol: TLSA. Tech. rep., IETF RFC 6698 (2012)
28. Holz, R., Riedmaier, T., Kammenhuber, N., Carle, G.: X. 509 forensics: Detecting and localising the SSL/TLS men-in-the-middle. In: 17th European Symposium on Research in Computer Security (ESORICS), pp. 217–234 (2012)
29. ICSI: The ICSI certificate notary (2011), https://notary.icsi.berkeley.edu/
30. Kasten, J., Wustrow, E., Halderman, J.: Cage: Taming certificate authorities by inferring restricted scopes. In: 17th Financial Cryptography and Data Security Conference (FC). pp. 329–337 (2013)
31. Kim, T., Huang, L., Perrig, A., Jackson, C., Gligor, V.: Accountable key infrastructure (AKI): A proposal for a public-key validation infrastructure. In: 22nd International Conference on World Wide Web (WWW). pp. 679–690 (2013)
32. Langley, A.: Public key pinning (May 2011), https://www.imperialviolet.org/2011/05/04/pinning.html
33. Langley, A.: Further improving digital certificate security (December 2013), https://security.googleblog.com/2013/12/further-improving-digital-certificate.html
34. Laurie, B., Kasper, E.: Revocation transparency (2012), http://sump2.links.org/files/RevocationTransparency.pdf
35. Laurie, B., Langley, A., Kasper, E., Google: Certificate transparency. Tech. rep., IETF RFC 6962 (2014)
36. Lewison, K., Coralla, F.: Backing rich credentials with a blockchain PKI (2016), http://pomcor.com/techreports/BlockchainPKI.pdf
37. Liu, Y., Tome, W., Zhang, L., Choffnes, D., et al.: An end-to-end measurement of certificate revocation in the web's PKI. In: 15th Internet Measurement Conference (IMC). pp. 183–196 (2015)

38. Marlinspike, M.: Convergence (September 2011), https://github.com/moxie0/Convergence
39. Marlinspike, M.: Trust assertions for certificate keys. Tech. rep., IETF Internet-draft (2013)
40. Matsumoto, S., Reischuk, R.: IKP: Turning a PKI around with decentralized automated incentives. In: 38th IEEE Symposium on Security and Privacy (S&P) (2017)
41. Melara, M., Blankstein, A., Bonneau, J., Felten, E., Freedman, M.: CONIKS: Bringing key transparency to end users. In: 24th USENIX Conference on Security Symposium. pp. 383–398 (2015)
42. Micheloni, A., Fuchs, K., Herrmann, D., Federrath, H.: Laribus: Privacy-preserving detection of fake SSL certificates with a social P2P notary network. In: 8th International Conference on Availability, Reliability and Security (ARES). pp. 1–10 (2013)
43. of Michigan, U.: Censys (April 2016), https://censys.io/
44. Microsoft: MS01-017: Erroneous verisign-issued digital certificates pose spoofing hazard (March 2001), https://technet.microsoft.com/library/security/ms01-017
45. Morton, B.: Public announcements concerning the security advisory (January 2013), https://www.entrust.com/turktrust-unauthorized-ca-certificates
46. Morton, B.: More Google fraudulent certificates (July 2014), https://www.entrust.com/google-fraudulent-certificates/
47. Muneeb, A., Jude, N., Ryan, S., Michael, J.: Blockstack: A global naming and storage system secured by blockchains. In: 2016 USENIX Annual Technical Conference. pp. 181–194 (2016)
48. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008), https://bitcoin.org/bitcoin.pdf
49. Namecoin Team: Namecoin (2011), https://www.namecoin.org/
50. PSYC: Certificate patrol (2014), http://patrol.psyced.org/
51. Ryan, M.: Enhanced certificate transparency and end-to-end encrypted mail. In: 21st ISOC Network and Distributed System Security Symposium (NDSS) (2014)
52. Soghoian, C., Stamm, S.: Certified lies: Detecting and defeating government interception attacks against SSL (short paper). In: 15th Financial Cryptography and Data Security Conference (FC), pp. 250–259 (2012)
53. Sotirov, A., Stevens, M.: MD5 considered harmful today (December 2008), http://www.win.tue.nl/hashclash/rogue-ca/
54. SSL Shopper: SSL certificate for Mozilla.com issued without validation (December 2008), https://www.sslshopper.com/article-ssl-certificate-for-mozilla.com-issued-without-validation.html
55. Start Commercial (StartCom) Limited: Critical event report (December 2008), https://blog.startcom.org/wp-content/uploads/2009/01/ciritical-event-report-12-20-2008.pdf
56. Szalachowski, P., Matsumoto, S., Perrig, A.: Policert: Secure and flexible TLS certificate management. In: 21st ACM Conference on Computer and Communications Security (CCS). pp. 406–417 (2014)
57. Vandersloot, B., Amann, J., Bernhard, M., Durumeric, Z., et al.: Towards a complete view of the certificate ecosystem. In: 16th Internet Measurement Conference (IMC). pp. 543–549 (2016)
58. VASCO Data Security International Inc.: DigiNotar reports security incident (August 2011), https://www.vasco.com/about-vasco/press/2011/news_diginotar_reports_security_incident.html

59. Wendlandt, D., Andersen, D., Perrig, A.: Perspectives: Improving SSH-style host authentication with multi-path probing. In: 2008 USENIX Annual Technical Conference. pp. 321–334 (2008)
60. Wikipedia: Flame(malware) (March 2017), `https://en.wikipedia.org/wiki/Flame\_(malware)`
61. Wilson, K.: Distrusting new CNNIC certificates (April 2015), `https://blog.mozilla.org/security/2015/04/02/distrusting-new-cnnic-certificates/`
62. Zusman, M.: Criminal charges are not pursued: Hacking PKI (2009), `https://defcon.org/images/defcon-17/dc-17-presentations/defcon-17-zusman-hacking\_pki.pdf`

A Parameters Selection

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 watch for fraudulent certificates in the blockchain more frequently, and take countermeasures more quickly. Accordingly, we set $T_B = 120$ minutes as a typical value and let $N = 6$ (the same as the requirement in Bitcoin). In order to keep the block mining stable, the community adjusts the PoW target of the blockchain periodically.

The validity period of Type-I transactions (denote as T_I) is chosen to provide moderate revocation transparency. First, only when a transaction has been 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. Meanwhile, 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. So we require that $T_I \leq 10 \times T_{Revoke}$, where T_{Revoke} is the revocation status update period. For more than 95% of CRL files, T_{Revoke} is not larger than 1 day. OCSP provides timely revocation status services, but the validity period of OCSP responses is typically 4 or 7 days.[4] Thus, we set $T_I = 14,400$ minutes (or 10 days) in the prototype.

T_{II} determines the frequency of shadow Type-II transactions. We set $T_{II} = 10 \times T_I$ (i.e., 100 days).

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