Hindawi
Security and Communication Networks
Volume 2018, Article ID 8527010, 16 pages
https://doi.org/10.1155/2018/8527010

Research Article

Accountable and Transparent TLS Certificate Management: An Alternate Public-Key Infrastructure with Verifiable Trusted Parties

Salabat Khan,[1] Zijian Zhang,[1] Liehuang Zhu,[1] Meng Li,[1] Qamas Gul Khan Safi,[2] and Xiaobing Chen[1]

[1]*School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China*
[2]*University of Engineering and Technology, Taxila, Pakistan*

Correspondence should be addressed to Liehuang Zhu; liehuangz@bit.edu.cn

Received 4 February 2018; Accepted 10 June 2018; Published 18 July 2018

Academic Editor: Sherali Zeadally

Current Transport Layer Security (TLS) Public-Key Infrastructure (PKI) is a vast and complex system; it consists of processes, policies, and entities that are responsible for a secure certificate management process. Among them, Certificate Authority (CA) is the central and most trusted entity. However, recent compromises of CA result in the desire for some other secure and transparent alternative approaches. To distribute the trust and mitigate the threats and security issues of current PKI, publicly verifiable log-based approaches have been proposed. However, still, these schemes have vulnerabilities and inefficiency problems due to lack of specifying proper monitoring, data structure, and extra latency. We propose Accountable and Transparent TLS Certificate Management: an alternate Public-Key Infrastructure (PKI) with verifiable trusted parties (ATCM) that makes certificate management phases; certificate issuance, registration, revocation, and validation publicly verifiable. It also guarantees strong security by preventing man-in-middle-attack (MitM) when at least one entity is trusted out of all entities taking part in the protocol signing and verification. Accountable and Transparent TLS Certificate Management: an alternate Public-Key Infrastructure (PKI) with verifiable trusted parties (ATCM) can handle CA hierarchy and introduces an improved revocation system and revocation policy. We have compared our performance results with state-of-the-art log-based protocols. The performance results and evaluations show that it is feasible for practical use. Moreover, we have performed formal verification of our proposed protocol to verify its core security properties using Tamarin Prover.

1. Introduction

Transport Layer Security (TLS) is the backbone and grand success in securing network-based communication. Most of the financial and commercial applications as well as noncommercial applications depend on TLS for security. TLS provides defense against different kinds of attacks [1–3]. The current TLS Public-Key Infrastructure (PKI) is a huge and complex system consisting of processes, policies, and entities that are responsible for secure certificate management processes. It is essential to securely communicate and reliably link a public key with its owner [4–7]. All the processes, policies, and entities play their role in securing the TLS PKI, but CA is the primary trusted anchor and entity in the current PKI [8]. CA signs and issues certificates for the domain (Server). The certificate is considered trusted if trusted CA signs it. Therefore, CA is the primary anchor and party in PKI, where integrity and security of the current PKI depend on the security, trustworthiness, and reliability of CA. Unfortunately, in recent years many attacks have been launched against TLS PKI infrastructure, and several most tectonic attacks have revealed the vulnerability of CA in practice. In particular, if a CA gets compromised in the present trust model of PKI, it can issue a counterfeit certificate for any domain under its authority. Such maliciously issued inauthentic certificate can be used for an extended period without being noticed. These types of vulnerabilities are widely been recognized in the literature [9, 10].

CA has issued bogus certificates all over the world, including France [11], Turkey [12], USA [13, 14], the Netherlands [15], and China [16]. Even the Symantec CA (SCA) owning near about quarter of certificate market share [17] was found of issuing a spurious certificate for Google domain and almost 2500 fake certificates were issued for unregistered and real domains as part of a test [18, 19]. Due to the issuance of bogus certificates by CAs, man-in-middle-attack (MitM) has been launched against various famous sites such Google, Skype, Yahoo, and Microsoft Live [20, 21].

These fraudulent and bogus certificates were issued either due to misconfiguration, software error, operational error, and social engineering or due to government enforcement and compulsion [22–24]. For example, an unauthorized certificate for Google domain was issued [11, 25] due to human errors. Similarly, a government can also compel CAs to issue a rogue certificate for a domain to launch compelled certificate creation attack [22]. Likewise, the Comodo was hacked by an Iranian hacker and issued unauthorized certificates for various domains [13, 26].

Certificate revocation is another big problem that needs to be addressed. A study and survey showed CAs owned by three different companies had issued around 75% out of all certificate. Only, Symantec CA (SCA) owned near about quarter of certificate market share, and GoDaddy CA had signed nearly 26% of all certificate in 2013 [17, 27–30]. Revoking GoDaddy certificate if its private key gets compromised would invalidate 26% of all HTTPS server certificates. This revocation will result in the unavailability of 26% of HTTPS servers. So revoking certificates of these significant CAs would result in substantial and collateral damage. In 2011, DigiNotar and Comodo got compromised [15, 31], the certificate of the former victim was revoked from browser CAs list [32], but the later victim certificate is still present in browser CAs list [33].

To solve these problems, some techniques have been proposed in the literature. Certificate transparency [34] is one such project that was initiated by Google for detecting the misbehavior of CA and refraining CA from fake certificate issuance by making certificate management process transparent. In this technique, public log implemented as Merkle hash tree is used as evidence. Each domain enrolls the CA-issued certificate on this log server. The server then returns the signed certificate time (SCT) to the domain, and the domain provides this SCT to a client on TLS connection setup as testament. This technique is not immune to attacks when CA get compromised. Policert [35] is another proposal that tries to empower domain by giving supremacy to a domain to describe their policy, certificate, and TLS connection setup properties. This scheme also uses public log server for validation, management, and enforcement of its policies. However, in this approach, no mechanism is specified to detect log misbehavior.

In our proposed protocol we have tried to solve the above problems and have contributed the following improvements.

(1) We explore two kinds of attacks for Policert and mitigate these attacks by introducing an improved revocation system and monitoring mechanism.

(2) The proposed scheme can handle the certificates hierarchy, intermediate CAs discovery, and provide a method to revoke CAs certificates without causing collateral damage.

(3) We have verified the core security properties of our proposed protocol using formal verification tool Tamarin Prover [36].

The rest of the paper is organized as follows. Section 2 describes related work. Section 3 gives an overview of preliminaries. Policert: Secure and Flexible TLS Certificate Management is summarized in Section 4. In Section 5, we overview in detail the proposed scheme. Section 6 highlights the security analysis. In Section 7, we present the performance and comparison and discuss some practical concerns in Section 8. Finally, Section 9 concludes the paper with remarks for future work.

2. Related Work

Numerous works have been done to address the security and trust issues of current TLS PKI [28, 37, 38]. Perspective [39], Convergence [40], and Observatory [41] are notary-based lightweight PKI, in which notary continuously monitors web server's certificates and stores these certificates in public repository. This public repository enables clients to confirm the server public key with that stored in public repository. Other approaches attempt to empower domain owners and limit the role CA [42, 43]. We discuss the proposals that use publicly variable log.

In 2011 EFF has started the Sovereign Key (SK) [44] project which is a public log-based approach for accountability of CA transaction. In the Sovereign Key technique, long time key is generated to get rid of browser certificate warnings. If a browser does not succeed in establishing a secure connection with the domain, it may result in hard failure. For signing TLS public key in Sovereign Key technique, each web server needs to have sovereign key pair logged on a server called time-line server. However, the main limitation of SK is that the clients have to rely only on the mirror of the server which does not provide efficiently verifiable proof that certificate is actually on the server. Another problem is that client need to query servers before connecting to the domain, introducing extra latency and sacrificing client privacy.

Certificate Transparency [34] is a technique that is proposed by Google to detect bogus certificate issued by a compromised CA to make certificate issuance transparent. It is an improvement over the Sovereign Key (SK) by organizing append-only log through Merkle hash tree. The log maintainer in CT can serve clients with two types of cryptographically verifiable proof: (I) proof that a given certificate resides on the log server; (II) proof that the log is an extension to the previous log. Since it uses Merkle hash tree for the log, so the proof generation and verification time are logarithmic. This CT technique has several limitations. First, CT can not efficiently prove that a specific certificate is not present in the log since certificates are stored in chronological order. Second, as the goal of CT was to detect CA misbehavior, so CT is vulnerable to attack when CA

gets compromised by an adversary to create and register fake certificates. However, CT does not guard against this attack by preventing clients from accepting these spurious credentials. Moreover, CT has no built-in revocation transparency, so revocation transparency called Revocation Transparency is proposed [45]. However, it is still not incorporated.

Accountable Key Infrastructure (AKI) [46] is public log-based technique that tries to enhance and improve certificate management transparency. AKI protects domain and client from a single point of failure such as when CA or log get compromised. Besides, the trust is divided among various parties and check and balance method is used for detecting the misbehavior of the trusted entities. Moreover, to solve certificate revocation problem in the log, they uses Merkle hash tree that stores data lexicographically rather than chronologically. The AKI uses the validator for ensuring log consistency and detecting log misbehavior for securing clients against attacks. Again the AKI depends heavily on third-party agent called validator for security. The second problem is that the public log server stores only currently active, valid certificates and keeps no record of revoked certificates (purged from the public log). Checking for revoked certificates in public log is inefficient (linear) in AKI. Moreover, AKI does not support multidomain certificates.

In Distributed Transparent Key Infrastructure (DTKI) [47], Certificate Transparency (CT), Sovereign Key, and Accountable Key Infrastructure (AKI) are combined for certificate management without having any trusted monitor. DTKI uses the Sovereign Key concept of signing with a master key and then registering the certificate on a public log just like in CT and AKI. After successfully enrolling the CA-issued certificate on the public log, a client will accept the certificate present in the public log and validated by the domain owner. In DTKI, the client is responsible for verifying the integrity of public log and trusted validator is no more required. However, DTKI uses gossiping methods for synchronization of log status and contacting log server before every connection results in high latency. Hence, client privacy is violated because the log server knows about the client connection with the domain. Furthermore, in DTKI every one has to trust on the mapping log maintainer. DTKI has no mechanism for recovery from domain master key comprises.

The Design, Analysis, and Implementation of ARPKI: an Attack Resilient Public-Key Infrastructure [48] is public log-based Public-Key Infrastructure providing certificate related services and is an improvement over Accountable Key Infrastructure. In ARPKI, registering certificate requires the domain to contact only one CA, but a domain designates n service providers. The entitled service providers perform cross-checking and monitoring of the each other in ARPKI. So ARPKI mitigates attacks when n-1 trusted parties start colluding with each other due to cross-checking and monitoring. The security parameters are also proved using Tamarin Prover. The first limitation is that protocol goes through cool-off period in case of key compromise or improper key update. The second drawback of ARPKI is extra latencies and client connection delay due to the involvement of the all trusted parties in all processes. Another problem is that the public log server stores only currently active, valid certificates and keeps no record of revoked certificates (purged from the public log). Checking for revoked certificates in public log is inefficient (linear) in ARPKI. Moreover, ARPKI does not support multidomain certificates.

In Policert: Secure and Flexible TLS Certificate Management [35], the domain owners are given more control over their certificate usage and verification by a specification of detailed subject certificate policy on the certificate usage. So a domain needs to describe and specify the properties of TLS connection. Policert is a public log-based scheme like Accountable Key Infrastructure AKI for management, publication, and enforcement of its policy. Multisignature certificate and subject certificate policy is inscribed on public log server. However, in this approach, the mechanisms for detecting and disseminating log misbehavior are unspecified. The second problem is that protocol goes through cool-off period in case of key compromise or improper key update. Another problem is that the public log server stores only currently active valid certificates and keeps no record of revoked certificates (purged from the public log). Checking for revoked certificates in public log is inefficient (linear) due to Lex-Tree in Policert. Moreover, Policert has no solution for certificate chain (intermediate CAs discovery).

To incorporate a revocation system and its monitoring mechanism into CT, an attempt was made in [45, 49, 50]. These proposals try to address the revocation system problem left open by CT. The certificate revocation checking process in Revocation Transparency [45] is linear and inefficient. Certificate Issuance and Revocation Transparency [50] improved the certificate revocation checking operation by using tree called Lex-Tree. However, CIRT cannot handle key loss, and a domain needs to create a fresh identity. Recently, another proposal called PKI Safety Net (PKISN): Addressing the Too-Big-to-Be-Revoked Problem of the current TLS ecosystem was proposed [49], to address the revocation problem of the CT, and provides guidelines for lightweight monitoring. This proposal has no support for a multiplicity of log servers that would be needed for certificates. However, none of the above techniques are incorporated into CT yet. Moreover, none of the above discussed log-based schemes (CT, AKI, APRKI, DTKI, and Policert) have a solution for certificate chain (intermediate CAs discovery). Table 1 highlights the pros and cons of the log-based PKI schemes that aim to address security and trust issues.

3. Preliminaries

3.1. Merkle Hash Tree. Merkle hash tree is maintained in the form of a binary tree where data is stored in leaf-node [51]. The remarkable characteristics of the Merkle hash tree are that it can prove set-membership and nonmembership in an efficient and publicly verifiable way. These proofs are mainly the proof of presence, proof of absence, proof of extension, and proof of currency. In literature, publicly verifiable log data structures are extensively studied [34, 44, 50, 52–54]. Merkle hash tree can be maintained either in the form of chronological or in ordered data structure. The possible implementation of the chronological data structure

Table 1: Strengths and weaknesses of surveyed approaches. Entries underlined represent major disadvantage of the corresponding scheme.

	SK	*CT*	*AKI*	*ARPKI*	*DTKI*	*Policert*
MitM attack mitigation	√	×	√	√	√	√
Domain key recovery	√	√	√	√	×	√
Client connection privacy	×	√	√	√	×	√
Domain certificate revocation	√	×	√	√	√	√
Efficient certificate revocation proof	×	×	×	×	√	×
Intermediate CA discovery	×	×	×	×	×	×
CA certificate revocation	×	×	×	×	×	×

is Merkle-tree as proposed in [34, 50, 52, 55]. The order data structure can be stored as Merkle-tree organized as binary search tree as in [50, 53].

3.2. Bilinear-Map. Let G, G_T be a multiplicative cyclic group of order q, and g is the generator. A function $e : G \times G \longrightarrow G_t$ is bilinear pairing map if it has the following properties:

(1) **Bilinearity:** ∀ P and Q ∈ G, ∀ a ∈ $\mathbb{Z}_q$ and b ∈ $\mathbb{Z}_q$ $e(P^a, Q^b) = e(P, Q)^{ab}$

(2) **Nongeneracy:** ∀ g ∈ G, $g \neq 0 \Longrightarrow e(g, g) \neq 1$

(3) **Computability:** e must be efficiently computable

3.3. Accumulator. A cryptographic accumulator was first introduced by Benaloh and de Mare [56], who defined accumulator as a one-way hash function having the property of quasicommutative. A quasicommutative function can be defined as $\int : X \times Y \longrightarrow X$ such that

$$\int \left(\int (x, y_1), y_2 \right) = \int \left(\int (x, y_2), y_1 \right), \quad \forall x \in X,\ \forall y_1, y_2 \in Y \tag{1}$$

Accumulators are further extended and improved by [57–61]. Our proposed scheme uses bilinear-map based accumulator for proving membership and nonmembership for a certificate in the log server. Suppose we have an instance of bilinear pairing and a set X = $\{x_1, x_2, \ldots, x_n\}$, such that $\forall x_i \in \mathbb{Z}_q^*$. Let s be the trapdoor information from $\mathbb{Z}_q^*$. Then the accumulation value of X is

$$Acc(X) = g^{(x_1+s)(x_2+s)\cdots(x_n+s)} \tag{2}$$

The witness for element $x_i \in X$ is $W_{x_i,X} = g^{\prod_{x_j \in X : x_j \neq x} (x_j+s)}$, the value s is the secret key sk of accumulator, the set $\{g^{s^i} \parallel 0 \leq i \leq q\}$ is the public key pk, and the verifier having the pk and accumulator value can authenticate the membership witness by testing

$$e\left(W_{x_i,X}, g^{x_i} g^s\right) \stackrel{?}{=} e\left(Acc(X), g\right) \tag{3}$$

Since g and g^s are part of public key and are equivalent to $e(W_{x_i,X}^{(x_i+s)}) \stackrel{?}{=} Acc(X)$ mathematically, the nonmembership witness for element $y \notin X$ is a pair $\widehat{W}_{y,X} = (W_{y,X}, V_y)$, where $V_y = -\prod_{x_i \in X}(x_i - y) \bmod q$ and $W_y = g^{((\prod_{x_i \in X}(x_i+s))+V_y)/(y+s)} = g^{\hat{q}X(s)}$ for some polynomial $\hat{q}X(s)$ of degree n-1, uniquely defined by set X. The nonmembership witness can be proved by verifier by checking the equation, having public key and accurate accumulated value.

$$e\left(W_y, g^y g^s\right) \stackrel{?}{=} e\left(Acc(X) . g^{V_y}, g\right) \tag{4}$$

This accumulator is collision-free under the q-Strong Diffie-Hellman assumption.

Definition 1 (negligible function). A function $\int : \mathbb{Z}_{\geq 0} \longrightarrow \mathbb{R}$ is negligible if $\forall x \in \mathbb{R}_{\geq 0}$, there exists $n_0 \in \mathbb{Z}_{\geq 1}$ such that $\forall n \geq n_0$, we have $|\int(n)| < 1/n^x$.

Definition 2 (q-strong Diffie-Hellman assumption [12]). Let g be the generator of cyclic group G of prime order q and $k \in \mathbb{Z}_q^*$. Any probabilistic polynomial time algorithm A that is given set $\{g^{k^i} : 0 \leq i \leq p\}$ can find a pair $(x, g^{1/(x+k)}) \in \mathbb{Z}_q^* \times G$ with probability at most $O(1/q)$.

Hash function: finally, we will use a secure collision-resistant hash function.

Definition 3 (collision-resistant hash function). Hash function h is a collision-resistant hash function if it is hard to find for two messages m_1, m_2 such that $h(m_1) = h(m_2)$ and $m_1 \neq m_2$.

4. Overview of Policert: Secure and Flexible TLS Certificate Management

We overview the Policert: Secure and Flexible TLS Certificate Management [35] for two reasons: (1) ATCM is inspired by Policert's design and concepts and employs some of its ideas and concepts; (2) ATCM addresses several shortcomings and limitations of it that we have identified. It was proposed to mitigate and protect domain and client from losses and vulnerabilities caused by CAs private key compromises [62, 63]. It works with the following five agents:

(1) Certificate Authority (CA): a CA authenticates domains and issues policy and X.509 certificates.

(2) Domain: a domain specifies the fine-grained policy and binds multisignature certificates from different CAs by signing with policy private key.

(3) Clients: a client wants to communicate with a domain for using services securely.

(4) Auditor: an auditor monitors log server operations and detects misbehavior.

(5) Log Server: Log Server maintains a database in the form of Merkle hash tree that logs domain policies and certificates.

The last two agents are not present in traditional PKI but are most important in Policert. The auditor is responsible for monitoring the log server and detects log server misbehavior, while log server is used to make CAs accountable. In Policert, the domain owners are given more control over their certificate usage and validation by a specification of detailed subject certificate policy (SCP). So domains are able to specify the properties of the TLS connection. It separates policy and certificate from each other so each domain has policy and certificate key pair. Separating key pairs provides security and allow domains to have multiple certificates and one policy. All the messages and actions are mainly divided into three categories. In the first type, certificate and policy management is done. The domain registers a CAs signed SCP on log server (LS) and binds together certificates from different CAs by signing with a subject certificate policy private key to create a certificate named multisignature certificate (MSC). The MSC is then registered on LS for secure communication with clients. The second and last category is concerned with LS audit and MSC validation and verification, respectively.

4.1. Attacks on Policert. Unfortunately, Policert has several loopholes and vulnerabilities through which MitM can be launched. The MitM attack can be made on Policert protocol through improperly revoked MSC as there is no proper method for MSC revocation consistency and revoked MSC are merely purged from log server. The revoked MSC can be inscribed at log server that had not recorded the certificate previously. Once the revoked certificate is logged, it can be used till the validity of the certificate.

Impersonation attack can also be launched against the Policert by compromising the log server. In this case, the attacker can use a malicious certificate to launch the attack against clients. The log server can use the different version of the database for launching the attack against the targeted victims as Policert has no mechanism to guard and detect such type of attack; ignoring security alerts and warning by victims is common in practice [64–67].

4.2. Policert Weaknesses. It has several drawbacks and problems that need to be solved. First, in Policert, only currently active certificates are stored and keep no record of revoked certificates as revoked certificates are merely removed from log server. Checking for a revoked certificate is linear in time, i.e., inefficient.

Second, Subject Certificate Policy (SCP) is not well defined and need to determine some more parameters for the proper management of certificates revocation since SCP did not include parameters for certificates revocation. Moreover, in Policert, it is necessary to register the SCP on log server and provide the proof to a client for validation, which introduces extra communication overhead.

Third, Policert has supposed that all certificates are directly signed by root CAs, which is unusual in practice. So it ignores certificates chain and hierarchy and intermediate CAs. Policert has no method to revoke CA certificates.

Finally, it has several loopholes and vulnerabilities through which MitM attack can be launched as discussed in detail in section Attacks on Policert.

5. Accountable and Transparent TLS Certificate Management: An Alternate Public-Key Infrastructure (PKI) with Verifiable Trusted Parties (ATCM)

Accountable and Transparent TLS Certificate Management: an alternate Public-Key Infrastructure (PKI) with verifiable trusted parties (ATCM) is a Public-Key Infrastructure (PKI) for managing certificates, domain policies, and making trusted parties like CAs accountable to the public. ATCM provides a strong defense against attacks by introducing cross-check method to detect misbehaving entity. We base ATCM on a verifiable log and extends Policert described in preliminary. In ATCM, we provide an overview of the principal agents involved and its responsibilities:

(1) Clients: client (user browser) is an actor that wants to connect with a domain (server) securely.

(2) Domain: a domain or server is an entity whose services are used by a client, and the client wishes to have a secure connection with a domain. The domain has public-key certificates signed by CAs and one or more key pairs, and this signed certificate will be presented to the client (user browser) for identification during TLS handshakes.

(3) Auditor: an auditor is an entity whose duty is to fetch records from a log server periodically and performs verification that all of the records and information are correct. It enables a client to ensure that proof provided by the server is correct and valid.

(4) Certificate Authority (CA): CA is responsible for signing certificates for domain owners. Before signing the certificate for the domain, she verifies domain owner's identity and signs subject certificate policy and certificates after the verification. However, unlike today's CA, the capability of CA in our proposed protocol is limited since signing and issuance of a certificate from a CA are not enough to make certificate valid and convince client (user browsers) to accept the certificate. However, in contrast to Policert, the CA checks the log server for their misbehavior and signs the root of log server. It mitigates the MitM chances.

(5) Log Server (LS): the ATCM scheme has log servers that record all certificates and efficiently generate proofs that can be efficiently validated and verified. These proofs are mainly the proof of presence, proof

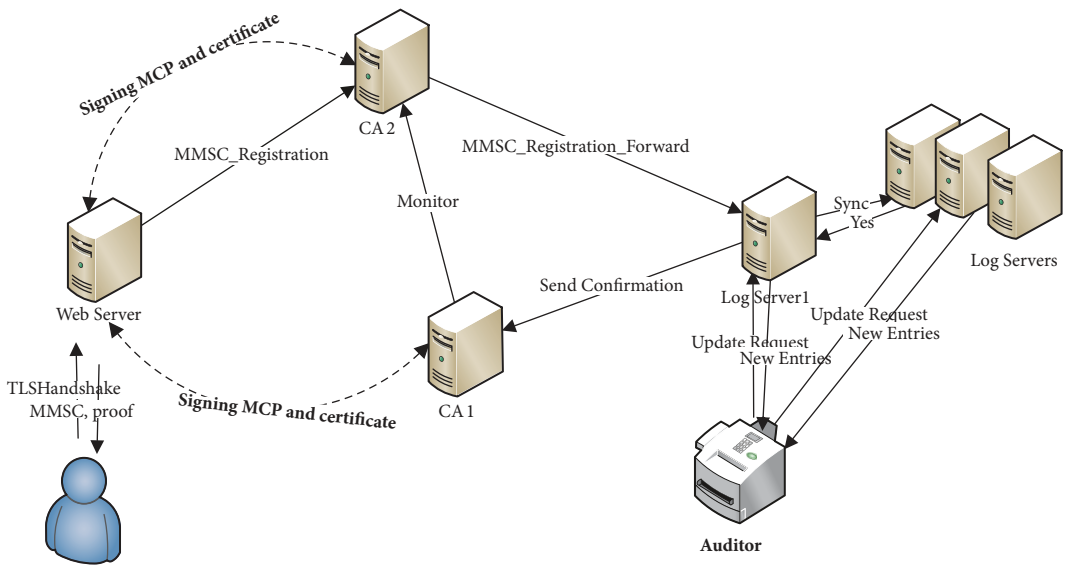

Figure 1: Accountable and Transparent TLS Certificate Management: an alternate Public-Key Infrastructure (PKI) with verifiable trusted parties (ATCM).

of absence, proof of extension, and proof of currency that are then used by clients, domain, and CAs. Log server can see all certificates globally; therefore, it can provide a global view and consistency checks. In ATCM, log server can also be compromised, and there is also check and balance on their actions and behavior.

The ATCM working is shown in Figure 1. In ATCM, three appointed entities are actively involved in supervising each other conduct and operation during an MMSC registration. Here CA1 validates the correctness and trustworthiness of the other two parties operations and perform the role of messenger among domain, log server1, and CA2, while CA2 acts as monitor ensuring that log server1, as well as other LSes, operates accordingly. The domain also designates log server1 to ensure that an MMSC is synchronized among all LS.

5.1. Architecture

5.1.1. Master-Key Signed Multisignature Certificates (MMSC). Master-key signed multisignature certificate (MMSC) binds a domain name to public key using more than one CA signatures. The MMSC is encoded as a collection of multiple standard x.509 certificates for backward compatibility with current PKI. For domain D MMSC is defined as follows with $n \geq 1$ CA signatures:

$$MMSC_D = \left(Cert_D^{CA1}, Cert_D^{CA2}, \ldots, Cert_D^{CAn}, Cert_D^{M_D}\right) \quad (5)$$

where $Cert_D^{CAi}$ represents a normal X.509 v3 certificate and $Cert_D^{M_D}$ represents policy binding of MMSC. In the proposed protocol, we also take into account the intermediate CAs hierarchy. The format of the multisignature certificate, when signed by intermediate CAs will be

$$\begin{aligned} MMSC_D = \Big(&Cert_{CA1}^{CA1} \longrightarrow Cert_{CAa}^{CA1} \\ &\longrightarrow Cert_D^{CAa}, Cert_{CA2}^{CA2} \longrightarrow Cert_{CAc}^{CA2} \\ &\longrightarrow Cert_D^{CAc}, \ldots, Cert_{CAn}^{CAn} \longrightarrow Cert_D^{CAn}, Cert_D^{M_D}\Big) \end{aligned} \quad (6)$$

The certificate $Cert_{CA1}^{CA1} \longrightarrow Cert_{CAa}^{CA1} \longrightarrow Cert_D^{CAa}$ represents the certificate chain of CA1. The certificates $Cert_{CA1}^{CA1}$, $Cert_{CA2}^{CA2}$, and $Cert_{CAn}^{CAn}$ are the root CA certificates while $Cert_{CAa}^{CA1}$ and $Cert_{CAc}^{CA2}$ are intermediate CAs certificates.

5.1.2. Master-Key Certificate Policy (MCP). The MCP binds information regarding usage, validation, and revocation of certificates to the domain name. The parameters are encoded in MCP as an extension in X.509 standard certificates [68], and MCP must be signed by a threshold number of CAs for its validity. Unlike SCP in Policert, MCP does not need to register on log server. MCP contains the following fields:

(1) General parameters

(i) POLICY_VERSION: it indicates current master-key certificate policy version number.

(ii) CA_LIST: it indicates domain trusted CAs list for signing MMSC and MCP.

(iii) MIN_CA: it indicates the minimum number of CAs signatures on MMSC for certificate registration on log.

(iv) CA_TH: it indicates threshold number of CAs signatures on MMSC to valid and must be less than the number of CAs in CA_LIST.

(v) COP_MK_UNLINKED: cool-off period is applied if new certificate is not signed by MCP private key.

(vi) COP_CA_UNTRUSTED: cool-off period is applied if CA signs certificate not present in CA_LIST.

(2) Additional parameters

(i) ONLY_EV: this field specifies that only extended validation (EV) certificates can be combined in MMSC.
(ii) MAX_CC_LENGTH: it refers to maximum length of a certificate chain.
(iii) NO_WILD_CERT: wildcards certificates are not allowed in MMSC.

(3) Security parameters

(i) MIN_CERT_SEC: define an MMSC certificate min security level.
(ii) MIN_TLS_SEC: TLS negotiated parameters min security level.

(4) Revocation parameters

(i) DOMAIN_KEY: this specifies the master key of a domain.
(ii) ROOT_CA_List: root CAs can directly revoke their certificates and intermediate CA certificates from a given checkpoint in a log called revocation-time-stamp. For domain certificate revocation MIN_CA root CAs are needed.
(iii) INTER_CA: intermediate CAs can directly revoke their own certificates from a given checkpoint in a log called revocation time-stamp. For MMSC revocation, MIN_CA CAs are needed.

(5) Connection Failure parameters

(i) CERT_TH_FAIL: when an MMSC is invalid due to CERT_TH.
(ii) TLS_FAIL: if TLS connection security level is too low.
(iii) POL_VER_NOT_REC_FAIL: if policy version is old.
(iv) EXP_LOG_PROOF_FAIL: if log server proof is outdated.
(v) LOG_PROOF_FAIL: if log server proof is invalid.
(vi) OTHER_FAILURE: aLL other types of failures.

5.1.3. Log Server. The number of log server is small and operated by Internet Service Providers (ISPs), web browser vendors, famous domains, or any interested parties. In short, log server has enough memory and computing power to store certificates and generate proofs. These proofs are mainly the proof of presence, proof of absence, proof of extension, and proof of currency. These proofs make log server behavior transparent to the world, and no one will blindly trust log server. In literature, publicly verifiable data structure is extensively studied [34, 44, 50, 52–54]. According to our literature survey, no single data structure is sufficient to provide all proofs needed by our proposed scheme. In our construction, log server is organized by using the tree data structures: Cert-Tree (CerT), Chain-Tree (ChT), Lex-Tree (LT), and Bilinear-map Accumulator Tree (BAT). The first two trees are maintained in the form to reflect the hierarchical structure of CAs, Lex-Tree is stored in the form of binary search Merkle-tree, and Bilinear-map Accumulator Tree is maintained in the form of accumulation tree.

Cert-Tree (CerT). In Cert-Tree(CerT) all certificates are logged in chronological order, where the log stores MMSC, revocation (REV), the root of the Chain-Tree (ChT), accumulation value A of BAT, and the digest of Lex-Tree and acts as time-stamping service. In this tree, all certificates are appended to the right of the tree. When CA request for MMSC logging, it is added to the right of the certificate tree in chronological order.

Chain-Tree (ChT). The Chain-Tree (ChT) consists of various subtrees in the form of a forest of trees and is stored in lexicographical order that mirrors the hierarchical and chain structure of certificate chains. The ChT leaf $H_i = H(MMSC \parallel t_i)$ represents a certificate and its associated registration time-stamp t_i. It also saves the possible revocation messages of a key if revoked; otherwise it stores ø, when a certificate has no associated revocation message. The parent represents the root CA while associated children nodes represent certificates signed by the parent node. The subtree of the parent node may be ø, when it does not sign any certificate.

Lex-Tree (LT). This tree is stored in the form of binary search Merkle-tree, where data items related to domain D $(D, List_R(MMSC_D), List_A(MMSC_D))$ are stored in such a way that left-right traversal results in lexicographic order of domain D. The size of the Lists is bounded by a constant X in Lex-Tree and is maintained in first-in-first-out (FIFO) fashion. Unlike, Lex-Tree in CIRT, we keep two lists: one for active MMSC and the other for revoked MMSC.

Bilinear-map Accumulator Tree (BAT). This tree is organized as binary search tree storing only active MMSC of domain lexicographically. Let C be the set of active MMSC; the C is implemented as a BAT. Each node in BAT contains an $MMSC \in C$ and the membership witness W_{MMSC} of MMSC. The accumulation value Acc(C) is linked to a leaf-node of the Cert-Tree.

5.2. Protocol Description

5.2.1. Certificate Registration. A domain owner spawns an MMSC by binding multiple X.509 certificates from a list of trusted CAs by signing with MCP private key. The domain owner explicitly designates trusted entities for performing MMSC registration, namely, two or more CAs and one log

server. The domain owner will send an MMSC to one of the CA, and other CAs will monitor the behavior of other entities involved in the certificate registration process. Upon receiving the registration request from the CA, the log server first validates the multisignature certificate. If so, the LS will send back the following message to the CA when a quorum of LS agrees; to ensure a consistent view of domain D $MMSC_D$.

$$\text{sign}_k\left(H\left(MMSC_D\right), t_a, t_1, t_2, t_n\right) \tag{7}$$

The above is a time-stamped registration receipt containing the newly registered certificate and guaranteed time of presence in the log. The $MMSC_D$ will be online after time t_a; t_1, t_2, and t_n show the certificate chain presence time of corresponding CAs. During the update of log server, the $MMSC_D$ will be added to the log server; the public parameters (PPs) are updated, the following operations are performed on log trees.

(i) Adding $MMSC_D$ to BAT: let C be the set of active certificates and compute the accumulation value Acc' according to the new set $C' = C \cup MMSC_D$ $Acc' = Acc^{(MMSC_D+s)}$, where s is the trapdoor information, i.e., secret key. For each $c_i \in C$, the membership witness is updated as $W_i' = W_i^{(MMSC_D+s)}$. The BAT root $digBT'$ is updated correspondingly.

(ii) Adding $MMSC_D$ to Lex-Tree: search tree for the domain D, if it exists, then add the new $MMSC_D$ to the active certificate list $List_A$ for D. Otherwise, spawn a new node for D with two lists for current and revoked certificate. Add the $MMSC_D$ to active certificate list and initialize revoked certificate list with null value; insert the newly created node to Lex-Tree in lexicographic order. The Lex-Tree root $digLT'$ is updated correspondingly.

(iii) Adding $MMSC_D$ to Chain-Tree: search for the root CAs certificates corresponding to the certificate chains in $MMSC_D$ in the certificate tree and append $MMSC_D$ to the last of the corresponding chains as the last node with revocation and subtree set to ø. Update the root $digChT$ of the tree.

(iv) Adding $MMSC_D$ to Cert-Tree: add a new node consisting ($MMSC_D$, Acc', $digLT'$, and $digRevT$) to the right of the append-only tree. The new root of the Cert-Tree is updated as $digCertT'$. Additionally, all intermediate CAs certificates are appended as well (if the intermediate certificates are not already in the LS).

5.2.2. Proof Querying. Anyone can send a query to the log server for the evidence of an MMSC. However, auditors are special entities responsible for detecting LS misbehavior by querying proofs from LS. The possible proof depending on the query can be the proof of the presence of a certificate, the absence of a certificate, absence of a domain, currency, extension, and presence of certificate chain. Anyone can request these proof from the log server and check the log server misconduct. The auditors, as well as well CAs, are primarily responsible for detecting the transgression of the log server.

5.2.3. Proof Generation. Upon receiving a request for verification, the LS yields the corresponding query as follows:

(i) Proof of presence of an MMSC: search for an MMSC in the Lex-Tree. If a node for the domain D is present in the Lex-Tree, send the sequence of hashes forming authentication path.

(ii) Proof of absence of an MMSC: search for the MMSC in the BAT. If there is no MMSC in the BAT, then send the witness $\widehat{W}_M MSC = (W_{MMSC}, V_{MMSC})$ for nonmembership of an MMSC, where $V_{MMSC} = -\prod_{MMSC_i \in C}(MMSC_i - MMSC) \bmod q \in \mathbb{Z}_q^*$ and $W_{MMSC} = g^{((\prod_{MMSC_i \in C}(MMSC_i+s))+V_{MMSC})/(MMSC+s)} = g^{\widehat{q}C(s)}$.

(iii) Proof of absence of a domain: for domain D, the log server first locates the two neighbor node D_i and D_{i+1} that bracket D, namely, $D_i \le D \le D_{i+1}$ (in Lex-Tree). Next, the log server sends a proof that D_i is in position i in log server and that D_{i+1} is in position i+1 in log server.

(iv) Proof of extension: proving that the current certificate tree $CertT'$ extends the previous Cert-Tree $CertT$ can be done logarithmical in time and space, by sending at most one digest (hash) per level. If $CertT'$ is an extension of $CertT$, then $CertT$ is a subtree of the $CertT'$ tree.

(v) Proof of currency: for an MMSC currency evidence, search it in BAT. If the search is successful, then send the membership witness $W_{MMSC_i,C} = g^{\prod_{MMSC_j \in C: MMSC_j \neq MMSC_i}(MMSC_j+s)}$.

(vi) Proof of presence of certificate chain: search the Chain-Tree for the certificate chain and yield the presence proof, by providing all the necessary nodes. The size of proof is not quite logarithm but instead needs $O(m\log_2 n)$ number of nodes, where m is Chain-Tree height and n is the maximum number of entries in the Cert-Tree. However, in almost all cases the Chain-Tree (ChT) will be very small.

5.2.4. MMSC and Log Proof Validation and Verification. After determining the policies parameters, the client validates and verifies the MMSC by checking that all certificates in MMSC sign the same public key, issued for a right domain, expiry date, having current policy version number, and validating certificate chain path. The verification is performed as follows.

(i) Proof of presence of an MMSC: the authentication path consists of hashes, the verifier will calculate the root hash from the path, and if the root hash calculated is equal to Lex-Tree digLT, then it is accepted, otherwise rejected.

(ii) Proof of absence of an MMSC: given the witness $\widehat{W}_M MSC = (W_{MMSC}, V_{MMSC})$ for nonmembership of an MMSC, calculate the $e(W_{MMSC}, g^{MMSC}, g^s) \stackrel{?}{=} e(Acc(C).g^{V_{MMSC}}, g)$ and accept if the equation holds, otherwise failure.

(iii) Proof of absence of a domain: a verifier receives a proof that D_i is in position i in log server and that D_{i+1} is in position i+1 in log server. The verifier can verify that these two nodes are neighbor on the same search path and that $D_i \le D \le D_{i+1}$, and this proves that D does not exist in Lex-Tree.

(iv) Proof of extension: the verifier computes that if $CertT'$ is an extension of $CertT$, then $CertT$ is a subtree of the $CertT'$ tree from the hashes sent by log server per level.

(v) Proof of currency: given the witness $W_{MMSC_i,C} = g^{\prod_{MMSC_j \in C: MMSC_j \neq MMSC_i} (MMSC_j + s)}$ for membership of an MMSC, calculate the $e(W_{MMSC_i,C}, g^{MMSC_i}, g^s) \stackrel{?}{=} e(Acc(C), g)$, and accepts if the equation holds, otherwise failure.

(vi) Proof of presence of certificate chain: the proof consists of hashes, the verifier will compute the root hash from the path, and if the root hash calculated is equal to Chain-Tree digChT then it is accepted, otherwise rejected.

5.2.5. MMSC Revocation. The revocation power of MMSC is only with a threshold number of CA. For revocation of an MMSC, domain owner designates the trusted agents as in registration process to have consistency. The revocation request consisting of an MMSC (revoking certificate) and appropriate signature is sent to responsible CA.

$$\begin{aligned} REV_{MMSC_D} &= \text{sign}_D\left(H\left(MMSC_D\right), revoke\right), \\ &\text{sign}_{CA1}\left(H\left(MMSC_D\right), revoke\right), \\ &\text{sign}_{CA2}\left(H\left(MMSC_D\right), revoke\right), \ldots, \\ &\text{sign}_{CAn}\left(H\left(MMSC_D\right), revoke\right) \end{aligned} \tag{8}$$

Let an MMSC be signed by two CAs; then the following message is sent to log server:

$$\begin{aligned} REV_{MMSC_D} &= \text{sign}_D\left(H\left(MMSC_D\right), revoke\right), \\ &\text{sign}_{CA1}\left(H\left(MMSC_D\right), revoke\right), \\ &\text{sign}_{CA2}\left(H\left(MMSC_D\right), revoke\right) \end{aligned} \tag{9}$$

As revoking CA certificate causes a substantial and collateral damage, so to avoid collateral damage caused by CA certificate revocation, it is performed from a specific checkpoint in the log server using time-stamp by the following message:

$$REV_{C_{CA}} = \text{sign}_k\left(H\left(C_{CA}\right), revoke\_from\_time\right) \tag{10}$$

The key k can be root CA key or intermediate CA key to revoke a certificate. The root CA can directly revoke its own, intermediate CA certificate from a certain point in the log server, while intermediate CA can revoke its certificate from a checkpoint in log server. The revocation message carries a revocation time-stamp that represents the specified time from which the specified CA certificate and revocation issuance must be invalidated. The log server processes the certificate revocation request upon receiving and returns a revocation receipt containing the certificate and guaranteed time of certificate revocation in the log server. The following operations are performed on the log server, and public parameters are also updated.

(i) Removing an MMSC from BAT: calculate the new accumulation value Acc' for new set $C' = C \setminus \{MMSC\}$ as $Acc' = Acc^{1/(MMSC+s)}$. The BAT is updated.

(ii) Revoking an MMSC in Lex-Tree: in the Lex-Tree, an MMSC is added to revoked certificate list for the corresponding domain.

(iii) Revoking an MMSC in Chain-Tree: only revocation message is added to the revoked MMSC and tree new root digest is calculated.

(iv) Revoking an MMSC in Cert-Tree: search an MMSC and append a revocation message to the corresponding certificate and the root digest is just updated by appending all trees new root digest to the rightmost node.

(v) Revoking CA certificate: to invalidate CA certificate from a checkpoint, append a revocation message to the chain containing the certificate to be revoked which will invalidate all the subsequent domain MMSCs from that time and onward. Moreover, a revocation message is also appended to the CA certificate in the Cert-Tree. The BAT and Lex-Tree are updated by invalidating all MMSCs effected by revoked CA certificate.

5.2.6. Client Connection to Domain. After completing the MMSC initial registration process, the domain D has an MCP, an MMSC, and its registration receipt or proof from log server. When a client initiates a TLS connection to a domain, the domain will send latest master-key certificate policy version and policy, master-key signed multisignature certificate (MMSC), and logging proofs for MMSC showing that the certificate is valid and related to the policy. After getting all these, the user browser will perform all the necessary actions and validates the certificates and proofs. The user browser will accept the connection if all operations are successful.

6. Security Analysis

Proof Using Tamarin. We have analyzed the security of our proposed protocol using formal protocol verification tool Tamarin Prover [36]. The Tamarin Prover is a symbolic tool for verification of security protocol that supports both unbounded verification and falsification in the nominal model. In the implementation of our model, we have abstracted several ideas just as in [47, 48, 69]. The log server is abstracted in the form of a list, and we also have only verified the security-related parameters of MCP instead of all other irrelevant information. In Tamarin Prover the protocols are coded as multiset rewriting rules while properties are

expressed in the form of first-order logic rules. The following shows the MMSC creation rule:

```
rule D_Create_MMSC:
let
    Policy = $ LSk + $ CA1 + $ CA2
    TD = time(~t)
    pkD = pk(~ltkD)
    pkPol = pk(~ltkPol)
in
[ !Ltk($D, ~ltkD), !Ltk( $D, ~ltkPol), F_CERT($D,pkD),
PublicFrVal(~t) ]
- -[ Is_Kind('Agent',$D), Is_Kind('LS', $LSk),
Is_Kind('CA',$CA1), Is_Kind('CA',$CA2),
MMSC_Req($D, ~ltkD) ,Clock(TD) ] ->
[ CombineMultipleCerts($D, sig(sig( (< 'Cert', $D, pkD,
pkPol, Policy >), ~ltkD) , ~ltkPol)),
Out(<$D, $CA1, sig( (< 'ReqForSigning', sig(sig( (< 'Cert',
$D, pkD, pkPol, Policy >) , ~ltkD), ~ltkPol), TD >),
~ltkD)>)
, Out(<$D, $CA2, sig( (< 'ReqForSigning', sig(sig( (<
'Cert', $D, pkD, pkPol, Policy >) , ~ltkD) , ~ltkPol), TD
>), ~ltkD)>)]
```

The adversary interacts with the protocols in Tamarin Prover by updating and generating network messages. The default adversary model used in Tamarin is a Delov-Yao adversary model, in which the adversary has control over the whole network and can intercept, delete, inject, and modify the network data and messages. We state our security goals in the form of implication in Tamarin syntax marked as a lemma. The following is one such goal to check the proper working of the protocol communication.

```
lemma Protocol_Proper_Work_Check:
"
(
All connid D B VL m key #i1.
    ( Communication ( connid, D, B, VL, m, key) @ #i1
    & not (Ex #i2 CA1 ltkCAx1.
        Compromise_CA1 ( CA1 , ltkCAx1 ) @ #i2)
    & not ( Ex #i3 CA2 ltkCAx2.
        Compromise_CA2 ( CA2, ltkCAx2 ) @ #i3)
    & not ( Ex #i4 K ltkK.
        Compromise_Ls (K, ltkK) @ #i4)
)
    ==>
    (
    not (Ex #i5. KU(m) @ #i5)
    )
"
```

In the lemma, a message is sent from domain D to user browser B when no party is compromised in the protocol to check the proper working of the protocol.

Analysis. Using the Tamarin, we have first checked the proper working of the protocol, then we have identified various expected attacks by compromising one, two, and all parties in the protocol. We find that our protocol can guard against attacks when at least one entity is not compromised by an adversary and verify the lemma but fail when the adversary controls all the entities involved.

We experimented the proof on a personal computer with an Intel (R) Core (TM) i7-4790 CPU (3.60GHz) and 8GB of RAM. We have run the proof on VMware Workstation 12Pro and Ubuntu 16.04 64 bits as an operating system.

7. Performance Evaluation

To assess the effectiveness and efficiency of Accountable and Transparent Public-key Infrastructure, we compare ATCM with existing log-based schemes CT [34], AKI [46], DTKI [47], Policert [35], and ARPKI [48].

7.1. Asymptotic Analysis. Let k, m, and n be the total number of the domain, active certificates, and certificates in log server, respectively, and X be the accumulation set of certificates in the log server. The asymptotic costs of the different schemes are shown in Table 2.

7.1.1. Numerical Analysis. Currently, there are $3.31 * 10^8$ domains [70], so we suppose the database is required to store certificates for $3.31 * 10^8$ domain, who enroll with the LS over five-year period. We also assume that, on average, 10% of the certificates are revoked per year. This amounts to 180,000 certificates issuance per day and 95,000 certificates revocations per day during five years, a total of 275,000 operations per day. Insertion and revocation occur in the order of $\log_2 3.31 * 10^8 \approx 29$ transactions on Lex-Tree, Cert-Tree, and Chain-Tree. This takes negligible time. However, the BAT takes O(m) operations as each element membership value is updated. But, we get constant proof and verification cost.

ATCM induces no extra network request (no extra round-trip latencies) to the TLS handshake. However, ATCM expands the TLS handshake message size by approximately a kilobytes due to LS proof stapling. To assess the log server proof size of various schemes, we set $n = m = k, n = m = 2*k$, $n = m = 4 * k$, $n = m = 8 * k$, $n = m = 16 * k$, and $n = m = 32 * k$, as usually domain can have many certificates, where k is number of domains, n is number of certificates, and m is currently active certificates (though only a fraction of domains have certificates and some log-based schemes do not support multidomain certificate (AKI, ARPKI)). We have considered worst cases where a total number of active certificates and issued certificates are equal.

Certificate Presence Proof. The first graph in the Figure 2 illustrates the certificate presence proof size in bytes. The proof cost in proposed schemes, AKI, ARPKI, and DTKI is $\log k$ and is independent of the number of certificates, so they have constant proof size. In CT and Policert, the authentication path size grows with the increase in the number of certificates while in ATCM, AKI, ARPKI, and

Table 2: Asymptotic Analysis of Log-based Schemes. Here, n is total number of certificate present in the log server, k presents total number of domains, and m represents total number of active certificates for domains in the log server.

Schemes	Parameters	Certificate presence proof	Certificate absence proof	Extension proof	Currency proof
CT[34]	Proof size	O(logn)	×	O(logn)	×
	Computation cost	O(logn)	×	O(logn)	×
	Verification cost	O(logn)	×	O(logn)	×
AKI[46]	Proof size	O(logk)	O(logk)	×	O(logk)
	Computation cost	O(logn)	O(logn)	×	O(logn)
	Verification cost	O(logn)	O(logn)	×	O(logn)
ARPKI[48]	Proof size	O(logn)	O(logn)	×	O(logn)
	Computation cost	O(logn)	O(logn)	×	O(logn)
	Verification cost	O(logn)	O(logn)	×	O(logn)
Policert[35]	Proof size	O(logn)	O(logn)	×	O(logn)
	Computation cost	O(logn)	O(logn)	×	O(logn)
	Verification cost	O(logn)	O(logn)	×	O(logn)
DTKI[47]	Proof size	O(logn)	O(logn)	O(logn)	O(logk)
	Computation cost	O(logn)	O(logn)	O(logn)	O(logk)
	Verification cost	O(logn)	O(logn)	O(logn)	O(logk)
ATCM(Our Scheme)	Proof size	O(logk)	O(1)	O(logn)	O(1)
	Computation cost	O(logk)	O(m)	O(logn)	O(m)
	Verification cost	O(logk)	O(1)	O(logn)	O(1)

DTKI path size expands with an increase in the number of the domains. Among all schemes, Policert has the highest certificate presence cost due to sending extra data for policy presence on log server which costs dlogn, where d is the depth of policy tree. However, AKI and ARPKI do not support multiple domain certificate which is a major limitation (a domain can have many certificates).

Certificate Absence Proof. Figure 3 compares certificate absence proof of log-based schemes. Again our proposed scheme has a constant cost of 64 bytes, as the nonmembership witness size $\widehat{W}_{MMSC,X} = (W_{MMSC,X} \in G, V_{MMSC} \in \mathbb{Z}_q)$ is 64 bytes. CT and DTKI send $\log n$ data for proving an absence of a certificate. AKI and ARPKI require $\log k$ hashes, while Policert needs $d \log k + \log n$ nodes to prove an absence of a certificate. Anew, our framework takes advantage of accumulation tree constant cost and outstanding among all PKI schemes.

Log Tree Extension Proof. The proof that the current tree is an extension of the previous tree needs $\log n$ hash for all schemes except AKI, ARPKI, and Policert which cannot provide such proof due to the lexicographical structure of log tree. Figure 4 gives an overview of the cost of extension proof.

Currency Proof. Figure 5 depicts the currency cost of various schemes. The ATCM has a constant and smallest proof size that is equal to the size of membership witness in X $W_{MMSC} \in G$, which is 32 bytes. The size of authentication path of a tree used in AKI, ARPKI, and DTKI is $\log k$; Policert has again $d \log k + \log n$ cost and more expensive. However, CT log tree has no such utility to provide currency proof.

7.2. Theoretical Comparison. In Table 3 we have several parameters like certificates, audit log properties, security, deployability, and efficiency to compare the log-based public-key protocols. For the first parameter certificates, no approach accept the self-signed certificate, and all have built-in support for certificate revocation except CT.

In audit log properties, the proof of the presence of certificate in the log can be provided by all infrastructure. All schemes can give proof of the absence of certificate and proof of currency of log tree except CT, while AKI, ARPKI, and Policert are not capable of providing the proof of extension of log tree from the previous version.

In considering security as parameters for comparison, all infrastructure can prevent man-in-the-middle attack except CT. CT has no mechanism to handle such attacks as it was proposed to detect CA misbehavior. Similarly, the scale of compromising trusted entities for man-in-the-middle attack is low for CT and DTKI. Compromising only log server is enough to launch an attack against them, while AKI and Policet require compromising both. ATCM and ARPKI require all trusted entities to be compromised. Only ARPKI, DTKI, and ATCM protocol provide formal security proof for verification of their security properties. The CT and DTKI have no mechanism to handle when domain losses its private key, as DTKI supposes that domain key can never be compromised. Likewise, DTKI did not preserve client connection privacy, since client must contact log server

Table 3: Comparison of various log-based public-key infrastructures based on certificate, audit log, security, deployability, and efficiency metrics. Entries underlined indicate major disadvantages of the corresponding scheme.

	CT	*AKI*	*ARPKI*	*DTKI*	*Policert*	*ATCM*
Terminology						
Log	Log	Integrated Log Server	Integrated Log Server	Certificate Log Maintainer	Log Server	Log Server
Monitor	Monitor	Validator	Validator (Optional)	-	Auditor	Auditor
Certificates						
Self-signed Certificate Support	×	×	×	×	×	×
Certificate Revocation Support	×	√	√	√	√	√
Audit Log Properties						
Proof of Presence	√	√	√	√	√	√
Proof of Absence	×	√	√	√	√	√
Proof of Extension	√	×	×	√	×	√
Proof of Currency	×	√	√	√	√	√
Tree Type	Chron	Lex	Lex	Chron, Lex	Lex	Chron, Lex
Security						
MitM Attack	Detection	Prevention	Prevention	Prevention	Prevention	Prevention
Multi Domain Certificate	√	×	×	√	√	√
Certificate Revocation	×	√	√	√	√	√
Formal Security Proof	×	×	√	√	×	√
No. of compromised parties for MitM	1/1	2/3	*All*	1/1	2/3	*All*
Domain Key Recovery	×	√	√	×	√	√
Client Connection Privacy	√	√	√	×	√	√
Intermediate CA Discovery	×	×	×	×	×	√
CA Certificate Revocation	×	×	×	×	×	√
Deployability						
Client-side-changes Required	√	√	√	√	√	√
Server-side-changes Required	×	√	√	×	√	√
Efficiency						
TLS-con-setup Add. Bandwidth	Bytes	KB	KB	KB	KB	KB
TLS-con-setup Extra Latency	×	×	×	√	×	×
End User Additional Action	×	×	×	√	×	×

before connecting to a domain. Hence, the log server has information about client connection so violating client privacy. Similarly, all schemes supposed that root CA signs their certificate directly and did not consider certificate hierarchy (intermediate CA) except ATCM, which can also handle intermediate CAs certificates. Only ATCM can revoke CA certificate without causing massive and collateral damage.

We also take into account deployability evaluate and compare the log-based public-key protocols. All protocols require client-side-changes as well as server-side-changes except CT and DTKI. At last, we also investigate efficiency. In TLS connection setup, all protocols require sending extra data in kilobytes except CT which has overhead in bytes. Similarly, end-user additional actions are needed in DTKI for connection setup. Moreover, DTKI also introduces extra latency in connection setup.

8. Discussion

CAs in ATCM. Certificate authorities in ATCM are different than the CAs in current CA model as the trust in an individual is gallantly reduced in ATCM and are made strongly

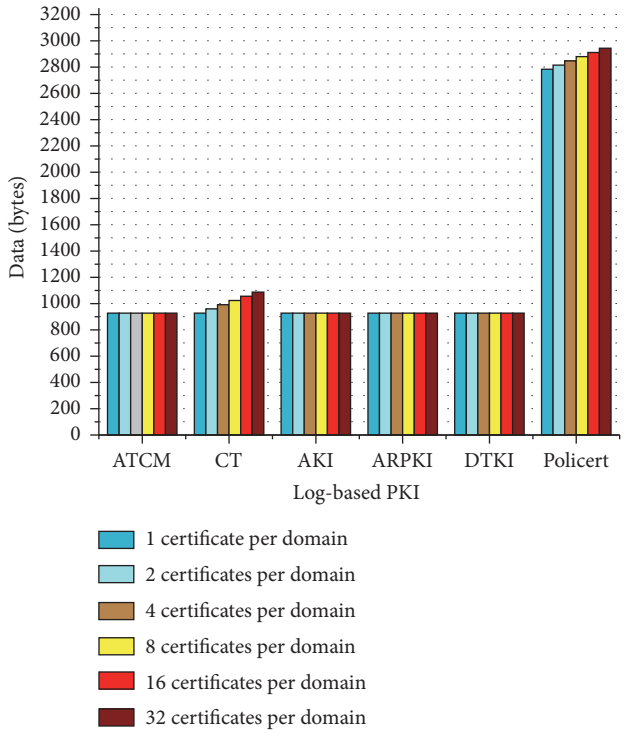

Figure 2: Certificate presence proof size.

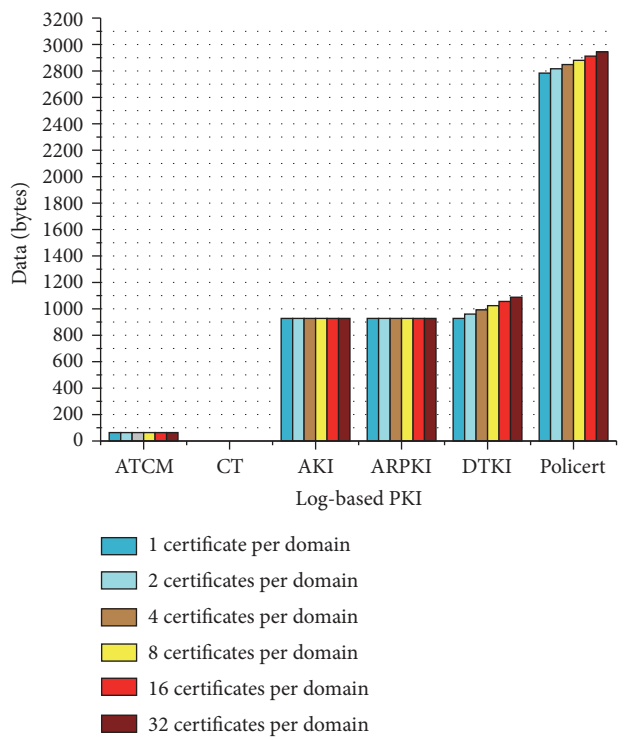

Figure 3: Certificate absence proof size.

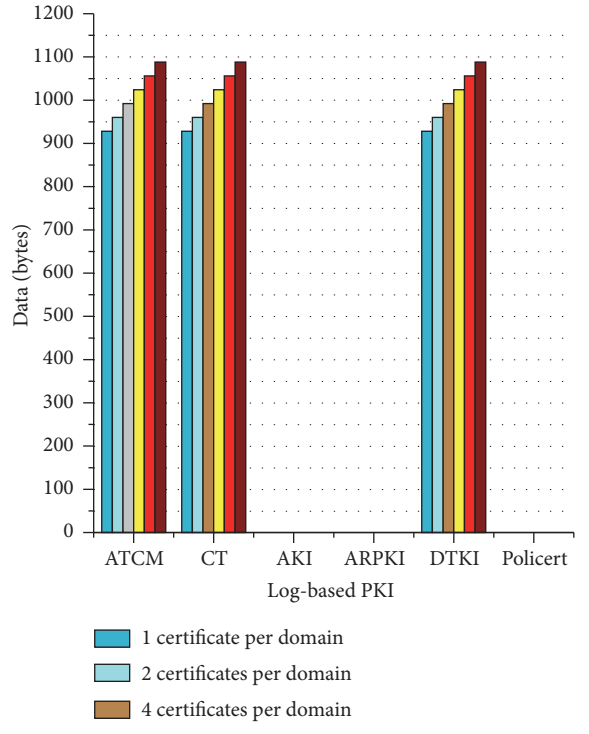

Figure 4: Extension proof size.

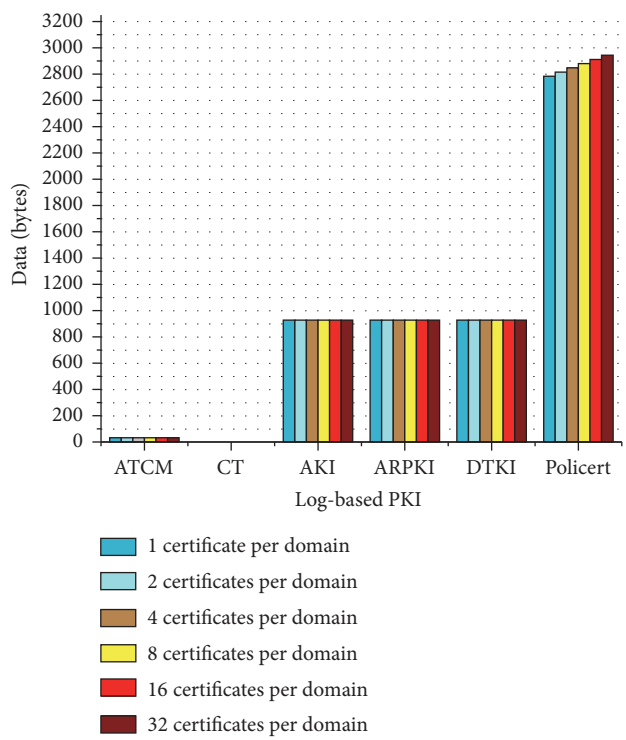

Figure 5: Currency proof size.

accountable to the public. Moreover, the LS operations are publicly visible and verifiable and that compelled and spurious certificates can be fast and easily detected.

Detection and prevention of successful MitM. ATCM provides strong security by preventing impersonation attack when at least one party is not colluding. However, in case all the parties start conspiring, then ATCM cannot defend clients from impersonation attack. ATCM can be extended to support the prevention and detection of such a powerful attacker using the two techniques outlined below.

Reactive approach. To detect the devious LS version attack, Gossip protocol is proposed [71], where clients randomly share signed root information about the LS with servers. This protocol enables clients to exchange information in a lightweight way and guarantee that they are viewing the same version of LS. Gossip protocol does not need extra infrastructure nor a dedicated connection as the messages are piggybacked over the original requests and responses. The only condition is some amount of servers and clients gossip. This approach can be directly applied to ATCM, making the detection of devious LS more robust.

Proactive approach. As Gossip protocol can detect the devious LS for misconduct but cannot prevent the clients from sustained MitM attacks; even the attack will remain undetected if the adversary controls victim's Internet access. To prevent clients from sustained MitM attacks, CoSi, a witness cosigning can used [72], which removes the need for Gossip protocol by guaranteeing that every certificate is witnessed and validated by a group of diverse witnesses before any client accepts it. This protocol protects clients against MitM attacks even if the adversary controls target's Internet access. This approach can be directly applied to ATCM, defending clients against sustained MitM attacks. However, this protocol will introduce latency and communication overhead.

9. Conclusion and Future Work

In this work, we presented Accountable and Transparent TLS Certificate Management: an alternate Public-Key Infrastructure (PKI) with verifiable trusted parties (ATCM), a new PKI which improves the security of current TLS PKI. ATCM provides high security so that it can prevent a MitM attack when only one entity out of all entities is entrusted. Even if all trusted parties involved in ATCM got compromised, in which circumstances MitM attack cannot be prevented, the other CAs or auditor may still get proof that of the compromise and can perform some countermeasure, even though MitM attack cannot be averted when all trusted parties get compromised but are at least apparent and visible.

The current certificate revocation system of all the log-based protocols is inefficient and has many problems. In ATCM protocol, we introduce a new revocation system, which removes several drawbacks of the current log-based system. ATCM has an improved domain policy, which adds a new and enhanced revocation policy so that CAs certificate can be revoked without causing collateral damage. This improved revocation system makes certificate revocation transparent and narrows the window of fake certificates to limit the range of attacks. Moreover, ATCM has a solution for intermediate CAs discovery and certificates chain validation and verification. Additionally, ATCM can handle certificate registration, certificate validation and verification, and certificate revocation in an apparent and transparent way.

To evaluate the performance and feasibility of ATCM, we have computed computational cost and communication overhead. The performance results and evaluation show that ATCM is feasible for practical use. Moreover, we have performed formal verification of ATCM protocol to verify its core security properties using Tamarin Prover. The formal verification shows that ATCM provides strong security against attacks.

In future, we plan to perform some experiments on industry adoption and interoperability with current TLS ecosystem. We also plan to improve MCP by exploring other additional parameters and study the impact of the parameters on the security and performance of the proposed protocol.

Data Availability

The supporting data and results generated during the current study are available from the corresponding author on reasonable request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is partially supported by China National Key Research and Development Program no. 2016YFB0800301 and National Natural Science Foundation of China "NSFC" no. 61300177.

References

[1] G. Aceto and A. Pescapé, "Internet Censorship detection: A survey," *Computer Networks*, vol. 83, pp. 381–421, 2015.

[2] M. A. Hussain, H. I. Jin, Z. A. Hussien, Z. A. Abduljabbar, S. H. Abbdal, and A. Ibrahim, "Enc-DNS-HTTP: Utilising DNS Infrastructure to Secure Web Browsing," *Security and Communication Networks*, vol. 2017, Article ID 9479476, 15 pages, 2017.

[3] A. Herzberg and A. Jbara, "Security and identification indicators for browsers against spoofing and phishing attacks," *ACM Transactions on Internet Technology (TOIT)*, vol. 8, no. 4, pp. 16:1–16:36, 2008.

[4] B. Qin, J. Huang, Q. Wang, X. Luo, B. Liang, and W. Shi, "Cecoin: A decentralized PKI mitigating MitM attacks," *Future Generation Computer Systems*, 2017.

[5] M. AlSabah, A. Tomescu, I. Lebedev, D. Serpanos, and S. Devadas, "PriviPK: Certificate-less and secure email communication," *Computers & Security*, vol. 70, pp. 1–15, 2017.

[6] Q. G. K. Safi, S. Luo, C. Wei, L. Pan, and G. Yan, "Cloud-based security and privacy-aware information dissemination over ubiquitous VANETs," *Computer Standards & Interfaces*, vol. 56, pp. 107–115, 2018.

[7] Q. G. Safi, S. Luo, L. Pan, W. Liu, and G. Yan, "Secure authentication framework for cloud-based toll payment message

dissemination over ubiquitous VANETs," *Pervasive and Mobile Computing*, vol. 48, pp. 43–58, 2018.

[8] A. S. Wazan, R. Laborde, D. W. Chadwick et al., "Trust management for public key infrastructures: Implementing the X.509 trust broker," *Security and Communication Networks*, vol. 2017, Article ID 6907146, 23 pages, 2017.

[9] S. B. Roosa and S. Schultze, "Trust darknet: Control and compromise in the internet's certificate authority model," *IEEE Internet Computing*, vol. 17, no. 3, pp. 18–25, 2013.

[10] A. Satapathy and J. Livingston, "A Comprehensive Survey on SSL/ TLS and their Vulnerabilities," *International Journal of Computer Applications*, vol. 153, no. 5, pp. 31–38, 2016.

[11] A. Langley, Further improving digital certificate security. (Decemeber 2013). URL http://googleonlinesecurity.blogspot.com/2013/12/further-improving-digital-certificate.html.

[12] A. Langley, Enhancing digital certificate security. (January 2013). URL http://googleonlinesecurity.blogspot.com/2013/01/enhancing-digital-certificate.html.

[13] Comodo: Comodo fraud incident 2011-03-23. (March 2011). URL http://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html

[14] Microsoft: Erroneous verisign-issued digital certificates pose spoofing hazard. (March 2001). URL http://technet.microsoft.com/library/security/ms01-017.

[15] Fox-IT, Black tulip: Report of the investigation into the diginotar certificates authority breach (August 2012). URL http://technet.microsoft.com/library/security/ms01-017.

[16] A. Langley, Enhancing digital certificate security. (March 2015). URL http://googleonlinesecurity.blogspot.com/2015/03/maintianing-digital-certificate-security.html.

[17] Usage of ssl certificate authorities for websites. (July 2016). URL https://w3techs.com/technologies/overview/ssl_certificate/all.

[18] R. Sleevi, Sustaining digital certificate security. (October 2015). URL https://googleonlinesecurity.blogspot.com/2015/10/sustaining-digital-certificate-security.html.

[19] S. Somogi and A. Eijdenberg, Improved digital certificate security. (April 2016). URL https://googleonlinesecurity.blogspot.com/2015/09/improved-digital-certificate-security.html.

[20] Microsoft: Improperly issued digital certificate could allow spoofing, Microsoft Security Advisory 3046310 (March 2015).

[21] E. Mills and D. McCullagh, Google, Yahoo, Skype targeted in attack linked to Iran. (March 2011). URL http://www.cnet.com/news/google-yahoo-skype-targeted-in-attack-linked-to-iran/.

[22] C. Soghoian and S. Stamm, *Certified lies: Detecting and defeating government interception attacks against SSL*, Springer, New York, NY, USA, 2012.

[23] M. Marlinspike, sslsni (July 2009). URL https://moxie.org/software/sslsniff/.

[24] M. Marlinspike, sslsni (December 2009). URL https://moxie.org/software/sslstrip/.

[25] ANSSI: Revocation of an igc/a branch (December 2013). URL http://www.ssi.gouv.fr/en/the-anssi/events/revocation-of-an-igc-a-branch-808.html.

[26] A. Niemann and J. Brendel, *A survey on ca compromises*, 2013.

[27] Domain name industry brief. Verisign. (2015). URL http://www.verisign.com/en_US/innovation/dnib/index.html.

[28] J. Clark and P. C. van Oorschot, "SoK: SSL and HTTPS: Revisiting Past Challenges and Evaluating Certificate Trust Model Enhancements," in *Proceedings of the 2013 IEEE Symposium on Security and Privacy (SP) Conference*, pp. 511–525, Berkeley, CA, USA, May 2013.

[29] A. Arnbak, H. Asghari, M. V. Eeten, and N. V. Eijk, "Security collapse in the HTTPS market," in *Queue*, ACM, New York, NY, USA, 2014.

[30] Z. Durumeric, J. Kasten, M. Bailey, and J. A. Halderman, "Analysis of the HTTPS certificate ecosystem," in *IMC*, ACM, New York, NY, USA, 2013.

[31] J. Appelbaum, *Tor Blog, Detecting Certificate Authority compromises and web browser collusion (22 March 2011)*, 2011.

[32] Mozilla Security Blog, DigiNotar removal follow up. URL https://blog.mozilla.org/security/2011/09/02/diginotar-removal-follow-up.

[33] Mozilla included CA certificate list (2015). URL https://wiki.mozilla.org/CA:IncludedCAs.

[34] B. Laurie, A. Langley, and E. Kasper, "Certificate Transparency," RFC Editor RFC6962, 2013.

[35] P. Szalachowski, S. Matsumoto, and A. Perrig, "Policert: Secure and Flexible TLS Management," in *Proceedings of the the 2014 ACM SIGSAC Conference*, pp. 406–417, Scottsdale, Ariz, USA, November 2014.

[36] S. Meier, B. Schmidt, C. Cremers, and D. Basin, "The Tamarin Prover for the Symbolic Analysis of Security Protocol," in *Proceedings of the 25th International Conference of Computer Aided Verification, CAV 2013*, vol. 8044 of LNCS, pp. 696–701, Springer, Princeton, NJ, USA, 2013.

[37] A. C. Grant, *Search for trust, An analysis and comparison of CA system alternatives and enhancements*, 2012.

[38] H. Tschofenig and E. Lear, Evolving the web public key infrastructure (2013). URL http://tools.ietf.org/id/draft-tschofenig-iab-webpki-evolution-01.html.

[39] D. Wendlandt, D. G. Andersen, and A. Perrig, "Perspectives: Improving ssh-style host authentication with multipath probing," in *Proceedings of the USENIX Annual Technical Conference*, R. Isaacs and Y. Zhou, Eds., pp. 321–334, USENIX Association, 2008.

[40] Convergence. URL http://convergence.io/.

[41] Electronic Frontier Foundation, SSL Observatory. URL https://www.eff.org/observatory.

[42] Public Key Pinning (May 2011). URL http://www.imperialviolet.org/2011/05/04/pinning.html.

[43] Public Key Pinning Extension for HTTP. (December 2011). URL http://tools.ietf.org/html/draft-ietf-websec-key-pinning-01.

[44] P. Eckersley, *Soveriegn key cryptography for internet domains, Internet draft*, 2012.

[45] B. Laurie and E. Kasper, Revocation Transparency. (2012). URL http://sump2.links.org/files/RevocationTransparency.pdf.

[46] T. H. J. Kim, L. S. Huang, A. Perrig, C. Jackson, and V. Gligor, "Acountable key infrastructure (AKI): A proposal for a public-key validation infrastructure," in *Proceeding of the 22nd international world wide web conference, ACM international world wide web conference steering committee*, pp. 679–690, Rio de Janeiro, Brazil, 2014.

[47] J. Yu, V. Cheval, and M. Ryan, "DTKI: A New Formalized PKI with Verifiable Trusted Parties," *The Computer Journal*, vol. 59, no. 11, pp. 1695–1713, 2016.

[48] D. Basin, C. Cremers, T. H. Kim, A. Perrig, R. Sasse, and P. Szalachowski, "Design, Analysis, and Implementation of ARPKI: An Attack-Resilient Public-Key Infrastructure," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 3, pp. 393–408, 2018.

[49] P. Szalachowski, L. Chuat, and A. Perrig, "PKI safety net (PKISN): Addressing the too-big-to-be-revoked problem of the TLS ecosystem," in *Proceedings of the 1st IEEE European Symposium on Security and Privacy, EURO S and P 2016*, pp. 407–422, Germany, March 2016.

[50] M. D. Ryan, "Enhanced Certificate Transparency and End-to-End Encrypted Mail," in *Proceeding of the 21st Annual Network and Distributed System Security Symposium (NDSS), The Internet Society*, 378, 369 pages, Springer, San Diego, CA, USA, 2014.

[51] R. C. Merkle, "A certified digital signature," in *Proceedings of the Conference on the Theory and Application of Cryptology*, pp. 218–238, Springer, 1989.

[52] R. C. Merkle, "A digital signature based on conventional encryption," in *Proceedings of the USENIX Secur. Symp*, pp. 369–378, Springer, Santa Barbara, CA, USA, 1987.

[53] A. Anagnostopoulos, M. T. Goodrich, and R. Tamassia, "Persistent authenticated dictionaies and their applications," in *Proceedings of the Information Security Conference*, pp. 379–393, Springer, Malaga, Spain, 2001.

[54] A. R. Yumerefendi and J. S. Chase, "Strong accountability for network storage," *ACM Transactions on Storage (TOS)*, vol. 3, no. 3, article 11, 2007.

[55] P. Maniatis and M. Baker, "Authenticated append-only skip lists," 2003, https://arxiv.org/abs/cs/0302010.

[56] J. Benaloh and M. De Mare, "One-way accumulators: A decentralized alternative to digital signatures," in *Proceedings of the Workshop on the Theory and Application of of Cryptographic Techniques*, pp. 274–285, Springer, 1993.

[57] N. Baric' and B. Pfitzmann, "Collision-free accumulators and fail-stop signature schemes without trees," in *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 480–494, 1997.

[58] J. Li, N. Li, and R. Xue, "Universal accumulators with efficient nonmembership proofs," in *Applied Cryptography and Network Security*, pp. 253–269, Springer, 2007.

[59] J. Camenisch and A. Lysyanskaya, "Dynamic accumulators and application to efficient revocation of anonymous credentials," in *Proceedings of the Annual International Cryptology Conference*, pp. 61–76, 2002.

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

[61] I. DamgÅrd and N. Triandopoulos, Supporting non-membership proofs with bilinear-map accumulators., IACR Cryptology ePrint Archive 2008 (2008) 538.

[62] P. Ducklin, The TURKTRUST SSL certificate fiasco what really happened, and what happens next? (January 2013). URL http://nakedsecurity.sophos.com/2013/01/08/the-turktrust-ssl-certificate-fiasco-what-happened-and-what-happens-next/.

[63] T. Sterling, Second firm warns of concern after Dutch hack (September 2011). URL http://news.yahoo.com/second-firm-warns-concern-dutch-hack-215940770.html.

[64] J. Sunshine, S. Egelman, H. Almuhimedi et al., "Crying wolf: An imperical study ssl warning effectiveness," in *Proceedings of the USENIX Secur. Symp.*, pp. 399–416, 2009.

[65] L.-S. Huang, A. Rice, E. Ellingsen, and C. Jackson, "Analyzing forged SSL certificates in the wild," in *Proceedings of the 35th IEEE Symposium on Security and Privacy, SP 2014*, pp. 83–97, May 2014.

[66] D. Akhawe and A. P. Felt, "Alice in warningland: A large-scale field study of browser security warning effectiveness," in *Proceedings of the USENIX security symposium*, vol. 13, 2013.

[67] A. P. Felt, R. W. Reeder, H. Almuhimedi, and S. Consolvo, "Experimenting at scale with google chrome's SSL warning," in *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems, CHI 2014*, pp. 2667–2670, Canada, May 2014.

[68] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile," Request for Comments 5280, 2008.

[69] J. Yu, M. Ryan, and C. Cramers, "How to detect unauthorized usage of key in," *Cryptology ePrint Archive*, 2015.

[70] Domain name industry brief. verisign (2017). URL http://www.verisign.com/en_US/innovation/dnib/index.xhtml.

[71] L. Chuat, P. Szalachowski, A. Perrig, B. Laurie, and E. Messeri, "Efficient gossip protocols for verifying the consistency of certificate logs," in *Proceedings of the 3rd IEEE International Conference on Communications and Network Security (CNS '15)*, pp. 415–423, IEEE, Florence, Italy, September 2015.

[72] E. Syta, I. Tamas, D. Visher et al., "Keeping Authorities "honest or Bust" with Decentralized Witness Cosigning," in *Proceedings of the 2016 IEEE Symposium on Security and Privacy, SP 2016*, pp. 526–545, USA, May 2016.