# X-FTPC: A Fine-Grained Trust Propagation Control Scheme for Cross-Certification Utilizing Certificate Transparency

Shushang Wen[1,3], Bingyu Li[2(✉)], Ziqiang Ma[4], Qianhong Wu[2], and Nenghai Yu[1]

[1] School of Cyber Science and Technology, University of Science and Technology of China, Hefei 230026, China
sswen@mail.ustc.edu.cn, ynh@ustc.edu.cn

[2] School of Cyber Science and Technology, Beihang University, Beijing 100191, China
{libingyu,qianhong.wu}@buaa.edu.cn

[3] Beijing Research Institute, University of Science and Technology of China, Beijing 100193, China

[4] School of Information Engineering, Ningxia University, Yinchuan 750021, China
maziqiang@nxu.edu.cn

**Abstract.** Cross-certification plays a fundamental role in facilitating the interconnection between different root stores in public key infrastructure (PKI). However, the existing trust management schemes (e.g., *certificate extension*) cannot implement fine-grained control over the trust propagation caused by cross-signing. This leads to the fact that although cross-certification expands the trust scope of certificate authorities (CAs), it also brings new security risks to the existing PKI system: (*a*) makes the certification path in PKI more complicated and lacks effective control, resulting in the arbitrary propagation of trust, and (*b*) more seriously, may even cause a revoked Cross-signed CA to continue to issue certificates that still have valid trust paths, due to the presence of cross-certificates that have not been fully revoked. Certificate Transparency (CT) is proposed to detect maliciously or mistakenly issued certificates and improve the accountability of CAs, by recording all certificates in publicly-visible logs. In this paper, we propose *X-FTPC*, a fine-grained trust propagation control enhancement scheme for cross-certification based on the idea of transparency, combined with the publicly-accessible, auditable, and append-only features of the CT log. *X-FTPC* introduces a new certificate extension to force the cross-signed CA

This work was supported in part by the National Natural Science Foundation of China under Grant 62002011, Grant 61772518, Grant 61932011, Grant 61972019, and Grant U21A20467; in part by the Youth Top Talent Support Program of Beihang University under Grant YWF-22-L-1272; in part by the China Postdoctoral Science Foundation under Grant 2021T140042 and Grant 2021M690304; in part by the Key RD Plan of Shandong Province, China under Grant 2020CXGC010115; and in part by the Beijing Natural Science Foundation through project M21031.


J. Lin and Q. Tang (Eds.): AC3 2022, LNICST 448, pp. 123–138, 2022.
https://doi.org/10.1007/978-3-031-17081-2_8

to submit an end-entity certificate to the specified log for pre-verification before it can be finally accepted. Fine-grained control of cross-certificate trust propagation is achieved through real-time monitoring of the certificate issuing behavior of cross-signed CAs. Moreover, it is fully compatible with CT frameworks that are widely deployed on the Internet.

**Keywords:** Public key infrastructure · Certificate transparency · Cross certification · Cross-signing · Trust management

## 1 Introduction

Traditional X.509 public key infrastructure (PKI) plays a fundamental role in establishing trust on the Internet. Based on digital certificates, PKI provides basic security services including authentication, confidentiality and data integrity in secure communications [5]. The application of the PKI system is based on the certificate authorities (CAs) being fully trusted and responsible for issuing certificates. The vendors of operating systems (OSes) and browsers evaluated the CA practices and then pre-installed a small group of CAs which located in the root stores in OSes or browsers, such that certificates signed by them or their subordinate authorities can pass validation [17,31].

However, due to the different security policies and service targets of different countries, regions or organizations, they usually trust only part of the secure and controllable root CAs according to their respective business needs. Currently, several mainstream OSes and browser vendors maintain their own public lists of trusted root CAs respectively [31], including Mozilla, Microsoft, and Apple. As a result, a specific CA organization may usually only be trusted by PKI users within a certain scope, and the trust cannot be propagated between various PKI user domains. Therefore, certificates issued by CA may not be trusted and accepted by all PKI users, and then formed separate islands of trust.

Cross-certification (also called cross-signing) [30] is proposed and widely used to alleviate the problem of trust islands, and the lengthy and costly validation processes that CAs must undergo to gain the trust of OSes and browsers [10]. That is, trusted CAs (typically called *Issuing CA*) can cross-sign other CAs (typically called *Cross-signed CA*) to extend their trust to them, and the resulting certificates typically called *cross-certificate*. Cross-certification provides a way for a certificate to obtain signatures from multiple issuers. That is, cross-certification creates several CA certificates that share the subject and public key, but each of them has a different issuer. Besides, it enables new CAs to quickly establish trust and also ensures extensive validation of certificates in the face of divergent root stores of OSes or browsers [10]. For example, several CAs, including Let's Encrypt [12] and GoDaddy [21], apply for cross-signing from other already trusted CAs before their own root certificates are included in root stores.

While cross-certification expands the trust scope of CAs, it also brings new security risks to the current PKI system: (*a*) makes the certification path in PKI more complicated [22] and lacks effective control, leading to the arbitrary propagation of trust. That is, once a cross-certificate is issued, it is no longer

restricted by the Issuing CA, but the credibility of any end-entity certificate issued by the cross-certificate is endorsed by the Issuing CA. A worse assumption is that a Cross-signed CA may arbitrarily expand its service scope and issue certificates for more applicants that are not anticipated by the Issuing CA, which will lead to the loss control of trust propagation during the cross-signing process; (*b*) may even cause a revoked CA certificate continues to issue certificates that still have valid trust paths, due to the presence of cross-certificates [13]. There are multiple trust paths for an end-entity certificate signed by a cross-signed CA, the trusted scope is greatly increased, and can be accepted as long as one of the trust anchors pointed to is trusted by the user. This extremely complicates alternative trust paths and increases the number of points for attack meanwhile. For example, suppose a root CA which has cross-signing relationships with other trusted root CAs is revoked, and then be removed from browser's root store. Theoretically, in this way, any certification path pointing to this CA will not be effective. However, if the browser does not strictly check the revocation status of the certificate, the revoked CA can still issue certificates trusted by browsers, and the certification paths of these certificates will eventually point to other trusted root CAs (e.g., DigiNotar in 2011 [13]) [10].

Unfortunately, the existing scheme for trust propagation control over cross-certification can only restrict trust propagation at a coarse-grained level, and cannot fundamentally solve the above problems. For example, certificate extensions including *Basic Constraints* extension, *Name Constraints* extension, etc. can only simply impose coarse-grained restrictions on the type, path length, and subject namespace of the certificate issued by a CA. Meanwhile, a Cross-signed CA can still issue a certificate that is not intended by the Issuing CA while meeting all of these certificate extension restrictions. Therefore, it's urgent for both browsers and CAs to construct a scheme which can control the trust propagation of cross-certificates in a fine-grained manner.

Certificate Transparency (CT) is proposed to detect fraudulent certificates and improve the accountability of CAs [8,14]. CT has been widely adopted by browsers and TLS software, including Chrome, Apple platforms, and Firefox, etc. In the CT framework, certificates are submitted to multiple public servers called logs by the CA that issues it. In response, the log generates a signed certificate timestamp (SCT), as the promise to make the certificate be publicly-visible in the logs, so that it is visible to monitors for further checking [17,18]. Then, the certificate is sent along with SCTs in TLS handshakes, otherwise, it will be rejected by CT-enabled browsers.

Inspired by the publicly-accessible and auditable features of CT, we propose *X-FTPC*, a CT-based fine-grained scheme for trust propagation control of cross-certificates and realize the enhancement of PKI certification at the same time. Through a new certificate extension, X-FTPC allows Issuing CA that issues cross-certificates to operate a designated CT log server (called *Mandatory-Log*), and can customize additional certificate verification criteria. The end-entity certificates issued by the Cross-signed CA must be submitted to the Mandatory-Log for recording and verifying. Only approved certificates can get the

*mandatory SCT* (called *m-SCT*) returned by the Mandatory-Log, otherwise it will be rejected by CT-enabled browsers in the TLS handshake. It should be noted that the end-entity certificate needs to be submitted to all mandatory logs that appear in the certification path (to obtain m-SCTs). If either m-SCT is missing, the end-entity certificate will be rejected by the CT-enabled browser.

In summary, X-FTPC provides cross-certification with the following enhancements:

- Fine-grained. Based on the certificate extension and CT log, a fine-grained cross-certificate trust propagation control scheme is realized, which can effectively provide the Issuing CA with the ability to restrict the issuance of any end-entity certificate by the Cross-signed CA.
- Revocable. By controlling the list of approved Cross-signed CAs in the Mandatory-Log, the restriction on the revoked CA and all related Cross-signed CAs from continuing to issue certificates with valid trust paths is realized.
- Easily-deployed. X-FTPC can be implemented and deployed by conveniently adding a certificate extension and based on the existing mature CT framework. Both CT and certificate extension have been widely adopted by browsers and TLS software.

The rest of this paper is organized as follows. The CT framework and the challenges with trust propagation control in PKI are reviewed in Sect. 2. The X-FTPC design details are described in Sect. 3. Section 4 presents the discussion of the feasibility in X-FTPC. Section 5 surveys the related work and Sect. 6 draws the conclusions.

## 2 Background and Challenges

In this section, we discuss how cross-certificates bootstrap the trust to root store, and analyse the shortcomings of certificate extensions for trust propagation control over cross-certificates in the Web PKI. Then, we briefly overview the certificate transparency, with a focus on the role of some key components.

### 2.1 Cross-Certification

In the trust model of Web PKI, CA acts as the trust anchor and holds self-signed certificates that are used to issue digital certificates to other entities. For flexibility and security considerations, root CAs typically delegate their signing capabilities by issuing intermediate CA certificates to their subordinate organizations, which in turn sign end-entity certificates to the website [31]. Browsers and OSes typically preinstall the public root trust list they maintain into the local root stores [17]. During the TLS handshake phase, the TLS server (e.g., website) sends the certificates to the TLS client (e.g., browser) to prove its identity. After receiving the certificates, the browser needs to construct a certification

path (also called *certificate chain*) and verifies that all certificates in the path are valid. For browser, a end-entity certificate passes verification only if it has a valid chain to a root certificate that exists in its local root store.

It takes time for a CA to be included in these root stores to satisfy the audit and certification process [10]. Before being accepted, in order to make the certificate issued by the *untrusted-CA* can be trusted by the browser, another *trusted-CA* whose root certificate is already contained in the root stores is usually used to cross-sign the root or intermediate certificate of the *untrusted-CA*, to create a trust path ending with the trusted root certificate of the *trusted-CA*. The certificate obtained through the above cross-signing is called cross-certificate.

As shown in Fig. 1, since root CA ($R_3$) is located outside the *Root Store*, the certificate ($E_6$) issued by intermediate CA ($I_5$) will not be trusted by browsers whose root stores as *Root Store*. Compare to $I_5$, $I_4$ is cross-signed by $I_3$. That is, $I_3$ issues a copy $I'_4$ which has the same subject name and public key with $I_4$. Therefore, both $I_4$ and $I'_4$ can verify the digital signature of $E_5$. Furthermore, since the root CA of $I_3$ ($R_2$) is stored in the root store, TLS client will accept the certificate $E_5$. In this way, the trust of cross-certificate is propagated to $E_5$. Similarly, CAs that are contained only in some root stores can use cross-signing to extend trust to further root stores. As a real-world example, Let's Encrypt has already issued a large number of certificates based on the cross-signing of IdenTrust before its own root was included in root store [12].

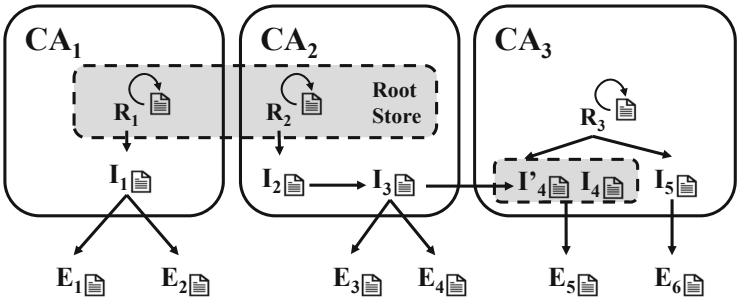


**Fig. 1.** An example of cross-signing in PKI system

### 2.2 Trust Propagation Challenge

Although the trust propagation control for cross-certification is a long-standing problem of traditional PKIs, until now there is no real perfect scheme to solve it from a fine-grained perspective.

In existing solutions, Issuing CA generally restricts the trust propagation scope and security level over cross-certificates through certificate extensions. For example, `Basic Constraints` as one of the most important extensions, although it works well in limiting the trust propagation scope, it only roughly

limits the length in certificate chain as well and cannot prevent the issuance of non-compliant certificates. `Policy Mapping` extension describes the mapping relationship of certificate policies between Issuing CA and Cross-signed CA. Subscribers on both sides can evaluate the security level of their certificates to each other based on this extension. While, `Name Constraints` extension restricts the namespace in both the `subject` field and `subject alternative name` extensions to achieve more specific control of the certificate issuing behavior of the subordinate CA. However, in real-world scenarios, both `Policy Mapping` and `Name Constraints` extensions are rarely contained in certificates [6,7] and client applications few validate these extensions during the validation process. As a real-world example, Swiss Government has two intermediate CAs cross-signed by *QuoVadis* and *Baltimore* respectively [10]. The one cross-signed by QuoVadis set the `Name Constraints` which whitelist domains that certificates are allowed to issue. However, the other cross-signed by Baltimore did not set the extension and 9 of 756 certificates were found that out of the whitelist domains. Besides, for compatibility, Apple's secure transport library does not support `Name Constraints` prior to OSX/macOS 10.13.3 [25].

Therefore, it is urgent to design a scheme which is more controllable and transparent, so as to achieve fine-grained control of trust propagation over cross-certificates in PKIs.

### 2.3 Certificate Transparency

CT is proposed against fraudulent certificates which bind a domain name to a key pair held by MitM attackers. As shown in Fig. 2, in the CT framework, some new components have been introduced and some components in the traditional PKI system have been enhanced with extra functions as follows.

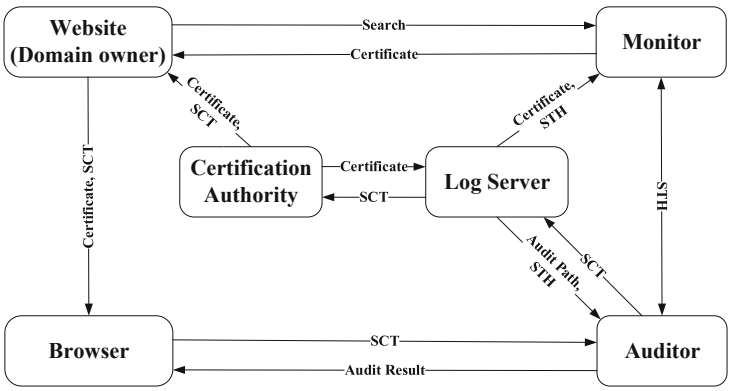


**Fig. 2.** The framework of certificate transparency

**Log Server.** A log is a public network service that records append-only certificates. Certificate is submitted by the CA to the log server, which responds with

a signed certificate timestamp (SCT), as the promise to make the certificate be publicly-visible in the logs. Certificates in a log are organized as a Merkle hash tree, and the root node of the tree is periodically signed, called the signed tree head (STH), to facilitate the audits.

**CA.** CA submits the signed certificate to the log server to obtain the SCT. Then, the SCT is delivered to the website server along with the certificate. Alternatively, before signing the certificate, the CA creates a *precertificate* that binds the same data in a different format as the final certificate. Then, the precertificate is submitted to the log to return an SCT, and the SCT is embedded in the final certificate. According to the CT policy, a certificate may be submitted to multiple log servers to obtain multiple SCTs.

**Website.** In addition to the CA submitting the certificate to the log when issuing certificates, in the early stage of CT deployment, sometimes the website also needs to submit its own certificate to the log server to obtain the SCT by itself. Then, in the TLS handshake, the website sends SCTs along with the certificate to browsers [14], as the embedded certificate extensions or the TLS extensions (e.g., OCSP stapling).

**Browser.** In TLS handshakes, a CT-compliant browser verifies the certificate and SCTs based on the pre-installed public keys of CAs and approved log servers [9]. If the browser's CT policy is not met, for example, there are not enough valid SCTs, the browser will reject the certificate.

**Monitor.** Monitor is responsible for retrieving logs to find suspicious certificates issued by the CA incorrectly or maliciously. A monitor periodically obtains and decodes all records from the log, and then checks the certificate of interest.

**Auditor.** Auditor is responsible for auditing the compliance of the log server. The auditor periodically requests STHs from the log server to check whether the log is append-only and consistent. Besides, it also requests an audit path to check whether a certificate is recorded in the public log.

## 3 X-FTPC: Fine-Grained Trust Propagation Control for Cross-Certification

In this section we firstly give an overview of the X-FTPC. Then, we describe how the new-add components work and extend X-FTPC in different scenarios. Finally, we discuss the potential threats to browsers and websites with X-FTPC.

### 3.1 Overview

Our goal is to achieve fine-grained control over the trust propagation caused by cross-certificate. In this paper, we define the *fine-grained* as: for each end-entity certificate that the Cross-signed CA expects to issue, the Issuing CA can decide whether to approve the issuance. The core idea of X-FTPC is enabling the

Issuing CA can not only monitor the certificate issuing behavior of Cross-signed CA, just like the CT does, but also control whether it's permitted to issue. It's worth noting that the X-FTPC only works when cross-certificates exist in the end-entity certificate's certification path.

To do this, we design a new-add public log server which is called the *Mandatory-Log* (usually referred to as *M-Log* in this paper). The M-Log has the same storage structure (i.e., MerkleTree) and function (e.g., publicly auditable) as the CT log (usually referred to as *Regular-Log*). In particular, M-Log has the additional function of certificate verification which the specific verification criteria is customized by the Issuing CA. X-FTPC also borrows the spam control mechanism used in Regular-Log [14]. That is, each log holds an acceptable list of CAs and only accepts the (pre)certificates whose root CA belongs to the list [14,16]. But different from that in CT, each Issuing CA in X-FTPC operates its own M-Log and maintains a cross-signing list (called *X-List*) in the Log, which includes CA certificates cross-signed by the Issuing CA or the other CAs. In Sect. 3.2, we describe more details about the M-Log and the X-List.

For ease of understanding and highlighting the compatibility with CT, we take the common three-level certificate chain in the Web PKI as an example to describe the main process in X-FTPC as shown in Fig. 3.

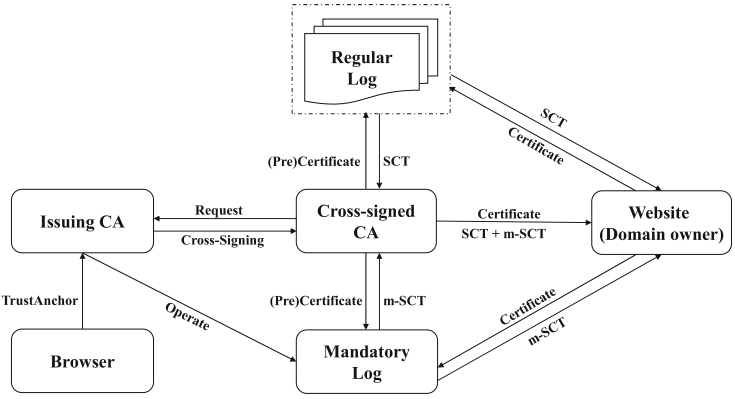


**Fig. 3.** The framework of X-FTPC

**Cross-Signing.** Issuing CA issues a cross-certificate with a new critical certificate extension (called `MandatoryLogIdentifier`) for the requested Cross-signed CA. Specifically, `MandatoryLogIdentifier` describes the information about the M-Log, e.g., gives its URL location in HTTP format. In Sect. 4, we'll discuss further about the security implications for browsers when this extension is set to "critical" or "non-critical". Then, Issuing CA adds the cross-certificate to the X-List in its M-Log actively. It is also important to note that if there are other cross-certificates present in the certification path, they should also request the Issuing CA to add them to the X-List.

**Issuing the Compliant End-Entity Certificates.** The SCT list of a compliant end-entity certificate in X-FTPC must have two types of SCT, which are returned by the Regular-Log and M-Log respectively. When the M-Log receives a submitted (pre)certificate chain, it checks whether the Cross-signed CA is included in its X-List and then verifies the end-entity certificate according to its verification criteria. The M-Log will return the m-SCT if and only if the above two requirements are met. Specifically, there are two methods for the Website to deliver the m-SCTs to browsers during a TLS handshake: (a) *TLS extensions.* After an end-entity certificate is issued, the Website or the Cross-signed CA submits it to logs to obtain an (m-)SCT and then delivers the SCT to browser as TLS extension. In this way, those Cross-signed CA issued certificates that have existed in the Internet in large quantities before the deployment of X-FTPC can still meet the policy requirements of X-FTPC; (b) *Certificate extensions.* Before an end-entity certificate is issued, Cross-signed CA submits a pre-certificate to the M-Logs to obtain m-SCTs. Then the certificate is issued with all the (m-)SCTs embedded as a certificate extension.

**Validation.** When browser validates the certificate chain sent by the websites or constructed by itself, it iterates all intermediate CA certificates so as to identify whether they have the `MandatoryLogIdentifier` extension. Then, browser validates whether the end-entity certificate contains m-SCTs returned by all the M-Logs given in the above extensions. After that, it further validates the other information in certificates according to the existing validation methods, e.g., the browser's CT policy.

### 3.2 Fine-Grained Control Based on Mandatory-Log in X-FTPC

In CT systems, each log holds an acceptable list of CAs and accepts only the (pre)certificates issued by these CAs [14,16]. Analogously, in X-FTPC, instead of the list of root CAs in Regular-Log, we propose the X-List in the M-Log, which is configured by the Issuing CA and includes CAs that are cross-signed by Issuing CA directly or indirectly. Here the indirect means that Cross-signed CA can continue to cross-sign other CAs to propagates trust further, which these CAs form an indirect cross-signing relationship with Issuing CA. Besides, the M-Log requires additional verification of the submitted certificate. Specifically, Issuing CA achieves the cross-certificate trust propagation control in fine-grained through the M-Log it operated from the following steps:

- Verifying the cross-certificates. For each certificate chain submitted to the M-Log, it verifies whether all CAs cross-signed by the Issuing CA directly or indirectly in the certificate chain are contained in the X-List. If one of the CAs is not in the X-list, then the end-entity certificates will be rejected for lack of the corresponding m-SCT.
- Verifying the end-entity certificate. Issuing CA customizes the additional verification criteria that the end-entity certificate must comply with.[1] If the end-

[1] In this paper, we do not restrict the specific format and content of the verification criteria. Issuing CA can define it according to their own application scenarios.

entity certificate fails to meet the criteria, it will be rejected and m-SCT will not be returned.

Combined with above, the M-Log in X-FTPC does need to do more work than Regular-Log in CT. On the other hand, however, the extra workload also solves part of the problems caused by the negligence of cross-certificates revocation (as mentioned in Sect. 1). In X-FTPC, when Issuing CA revokes a cross-certificate issued to a Cross-signed CA, it also removes the cross-certificate from the X-List in its M-Log. Since a X-FTPC-compliant certificate must contain all m-SCTs returned by the relevant M-Logs. Therefore, even if the others M-Logs in the certification path of the end-entity certificate that do not remove the revoked cross-certificates from their own X-List, it does not affect. In this way, X-FTPC enables browsers to avoid continuing to trust certificates issued by the revoked cross-signed CA as failure to perform the revocation checks.

In summary, Issuing CA centralizes the management of cross-certificates issued directly or indirectly through the X-List in M-Log and further achieves the fine-grained control of trust propagation by deciding whether to return the m-SCT.

### 3.3 Different Scenarios in X-FTPC

Based on the M-Log, we achieve fine-grained control of cross-certificate trust propagation in a high compatibility with CT meanwhile. Considering the different scenarios in practical application, we analyze the following scenarios, which demonstrate that X-FTPC can be easily extended. We take the CAs' trust relationships of Let's Encrypt [12] as a background for our scenario analysis. Let's Encrypt has already issued a large number of certificates based on the cross-signing of IdenTrust before their own root was included in the root store.

To begin with, as shown in Fig. 4, we assume that the following scenarios are based on the Web PKI and also make the definition as follows:

- $E_x$ refers to the browser, website or end-entity certificate depending on the context.
- $R_x$ is the trust anchor of $E_x$, which is already in-built in its root store, e.g., $R_1$ is the trust anchor of $E_1$ and $E_2$.
- $I_x$ are the intermediate certificates issued by the root CAs. $I'_x$ are the cross-certificates which cross-signed by another CAs, e.g., $R_1$ issues cross-certificates $I'_{R_2}$ and $I'_2$ for $R_2$ and $I_2$, respectively.
- $M - Log_x$ means that the M-Log is operated by x.

**Scenario 1 - Trust Propagation in Adjacent CAs.** In this scenario, as shown in Fig. 4, assumed that user (e.g., $E_1$) requests to connect with the website (e.g., $E_3$), and then $E_3$ sends a certificate chain to $E_1$. To validate the identification of $E_3$, there are two ways for $E_1$ to do it: ($a$) validate the certificate chain sent by $E_3$; ($b$) construct the certification path by itself. There exists three alternative certification paths: ($1$) $E_3 - I_2 - R_2$, ($2$) $E_3 - I'_2 - R_1$,

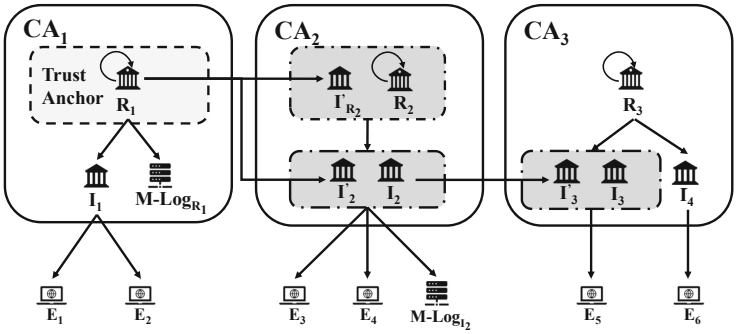


**Fig. 4.** Trust propagation among multiple CAs

and (*3*) $E_3 - I_2 - I'_{R_2} - R_1$. Since in Path (*1*), $R_2$ is out of the root store of $E_1$, so $E_3$ would not be accepted. In Path 2 and Path 3, as both $I'_2$ and $I'_{R_2}$ have the $MandatoryLogIdentifier$ extension which presents the information of $M - Log_{R_1}$, so $E_3$ must be submitted to the $M - Log_{R_1}$ for additional verification. But it's also noted that $I'_2$ and $I'_{R_2}$ are cross-signed by $R_1$ directly, so both of them are included in the X-List of $M - Log_{R_1}$ actively. Considering that user accepts $E_3$ only when the SCT list of $E_3$ includes the m-SCT from $M - Log_{R_1}$, so through this way, $R_1$ controls the trust propagation and certificate issuing behavior of $I_2$ in Path 2 and 3.

**Scenario 2 - Trust Propagation Across Multiple CAs.** Considering a more complicated scenario where there are multiple cross-signing between different CA domains along the certificate chain in Fig. 4. When the user (also $E_1$) communicates with $E_5$, it has the following alternative certification paths: (*1*) $E_5 - I_3 - R_3$, (*2*) $E_5 - I'_3 - I_2 - R_2$, (*3*) $E_5 - I'_3 - I'_2 - R_1$, and (*4*) $E_5 - I'_3 - I_2 - I'_{R_2} - R_1$. Different from the cross-signing directly in Scenario 1, $I_2$ also cross-signs $I_3$ and thus bootstraps the trust of $E_5$ to both $R_1$ and $R_2$ through $I'_3$, $I'_2$ and $I'_{R_2}$.

However, in Path 2, 3 and 4, as $I'_3$ has the $MandatoryLogIdentifier$ extension which points to $M - Log_{I_2}$, so $E_5$ also needs to be submitted to the $M - Log_{I_2}$ in addition to the $M - Log_{R_1}$ as Scenario 1 does. In particular, since $I_3$ is cross-signed by $R_1$ indirectly, $I'_3$ also needs to send a request to $R_1$ to be included in the X-List of $M - Log_{R_1}$. When $M - Log_{I_2}$ receives these certificate chains, it verifies whether the $I'_3$ is contained in its X-List and then verifies $E_5$ according to its verification criteria. The similar verification can be applied to $M - Log_{R_1}$. In Path 3, $M - Log_{R_1}$ verifies whether both $I'_2$ and $I'_3$ contained in its X-List. While in Path 4, $I'_{R_2}$ and $I'_3$ are verified. Therefore, even $I_3$ cross-signed by $R_1$ indirectly, $R_1$ also has the ability to control the certificate issuing behavior of $I_3$ (or $I'_3$) in Path 3 and 4.

**Scenario 3 - Internal Cross-Signing.** In above scenarios, we only discuss the cross-signing between different PKI domains, e.g., the issuer of $I'_{R_2}$ is attributed to $CA_1$ but the subject of $I'_{R_2}$ is attributed to $CA_2$. On the contrary, internal cross-signs are that the cross-certifcates' issuer and owner are attributed to the

same PKI domain [10], e.g., both the issuer and subject of $I'_3$ are attributed to $CA_2$, as shown in Fig. 5. We further discuss some of the differences between Scenario 1 and 2 when there exists internal cross-signs in Scenario 3. To begin with, we also assume that $E_1$ connects to the $E_5$ and then validates $E_5$ which has the following alternative certification paths: *(1)* $E_5 - I_4 - R_3$, *(2)* $E_5 - I'_4 - I_3 - R_2$, *(3)* $E_5 - I'_4 - I'_3 - I_2 - R_2$, and *(4)* $E_5 - I'_4 - I'_3 - I'_2 - I_1 - R_1$. Since $E_1$ only accepts $E_5$ in Path 4, so we do not discuss the other Paths here which their analysis are similar to the above scenarios. In Path 4, $E_5$ must be submitted to $M - Log_{I_1}$, $M - Log_{I_2}$ and $M - Log_{I_3}$ for additional verification.

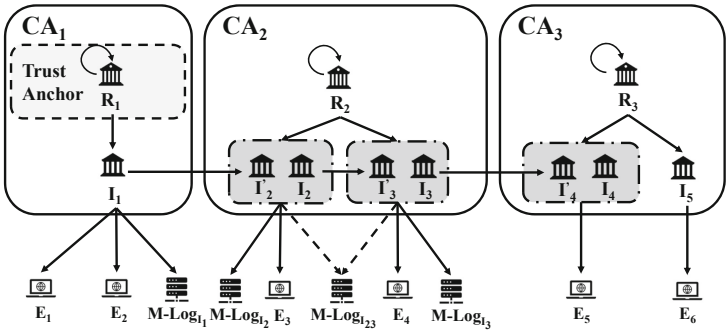


**Fig. 5.** Scenario of internal cross-signing

However, note that both $I_2$ and $I_3$ are governed within $CA_2$, we also propose that $I_2$ and $I_3$ can jointly operate an M-Log (called $M - Log_{I_{23}}$) to avoid resource redundancy and reduce the maintenance cost of CA. Correspondingly, whenever $I_2$ or $I_3$ cross-signs other CAs, only information about the $M - Log_{I_{23}}$ will be given in the $MandatoryLogIdentifier$ extension of the cross-certificate. For example, the $MandatoryLogIdentifier$ extension of $I'_4$ can only give the information about the $M - Log_{I_{23}}$. Then, $I_2$ or $I_3$ adds the cross-signed CAs to the X-List of $M - Log_{I_{23}}$ respectively. In particular, the X-List also contains all CAs which cross-signed internally, e.g., the $I'_3$. When $M - Log_{I_{23}}$ receives the certificate chain of Path 4, it can verifies whether $I'_3$ and $I'_4$ are included in X-List meanwhile. Most importantly, because of the relative independence between M-Logs, it does not weaken the fine-grained control of trust propagation by $I_1$. For example, $M - Log_{I_1}$ also needs to verify whether $I'_2$, $I'_3$ and $I'_4$ are included in its X-List.

### 3.4 Potential Threats in X-FTPC

Our primary goals in this paper are to design a scheme which can realize fine-grain control of cross-certificate trust propagation. Meanwhile, it can also provide sufficient security and will not introduce more potential threats than the existing CT. In particular, as X-FTPC is proposed based on CT, so it naturally inherits

the proven security attributes of CT [14]. Fundamentally, a misbehavior from compromised or misoperated CA against the X-FTPC usually results in either one of the two outcomes: (1) to issue a certificate which is not conform to the verification criteria (also unauthorized) in the M-Log; (2) to issue a certificate which is fraudulent but meeting the verification criteria. Briefly, we analyze the security in X-FTPC against the following threat assumptions.

**Case 1.** Cross-signed CA may collaborate with the website to issue a certificate that is outside the scope of its authority and does not submit the certificate to the corresponding M-Log. However, this certificate will not be accepted by the X-FTPC-enabled browser because it lacks the m-SCT returned by M-log. It's important to note that the X-FTPC only works if cross-certificates exist in the certification path. Thus, for a regular certificate chain without cross-certificates, the certificate without m-SCT can still be accepted by browsers.

**Case 2.** There is another case where Cross-signed CA may issue a fraudulent certificate which conforms to the verification criteria customized by the Issuing CA. When browsers validate the fraudulent certificate, it can be accepted as its SCT list meets the requirements stated in X-FTPC. Fortunately, X-FTPC also uses publicly-accessible logs as CT does, meaning that both auditors and monitors can independently verify whether the certificate really present in the M-Log. Besides, as the M-Log is monitored by Monitors just like the Regular-Log, thus the domain owner can detect the fraudulent certificate through the third-party monitor in time.

## 4 Discussion in Feasibility of X-FTPC

In this section, we discuss the feasibility of X-FTPC in CT-based PKIs around browsers. Essentially, how X-FTPC is deployed and the impact of its deployment is very similar to the CT framework [26]. An important measure of one scheme's feasibility is its compatibility with users (e.g., browsers). In X-FTPC, the SCT list of a website's certificate consists of two parts: (1) SCTs returned by the Regular-Log, (2) m-SCTs returned by the M-Log. With the $MandatoryLogIdentifier$ extension that provides information about the M-Log, browsers can be informed which m-SCT should be validated.

On the one hand, if the browser is required to enforce the X-FTPC verification (i.e., set $MandatoryLogIdentifier$ extension critical), the false positive rate may be too high in the early stage of X-FTPC deployment, that is, many valid certificates issued by cross-certificates will be rejected by the browser. On the other hand, if the browser is not required to enforce the X-FTPC verification (i.e., set $MandatoryLogIdentifier$ extension non-critical), then the scheme is weakened to the CT, which is fully compatible with the existing browser's CT policy, but a degradation attack may occur [19], resulting in a high false negative rate, that is, the browser will accept the "problem" certificate issued by the cross-certificate. Therefore, CAs and websites with high security requirements can implement mandatory checks by setting a white-list in the browser similar to

the CT mechanism [26]. For a wider scope of CAs and common domain names, we can achieve gradual deployment by setting deadlines.

In summary, as X-FTPC is based on the CT, so the deployment of it can be fully compatible and incrementally deployable. Besides, as X-FTPC-enabled browsers do not require much efforts to support its deployment, thus the feasibility and scalability are acceptable as well.

## 5 Related Work

With the abroad application of PKI, security researchers have studied a lot of work [1,3,4] around both CA ecosystem and certificate ecosystem. Among these works, there are few works involving the cross-signing or cross-certificate. As early as 2011, Holz et al. already noted the problem that once a untrusted CA is cross-signed by the other trusted CA which is contained in the root store may introduce heavy results [11]. Besides, the authors mentioned that the excessive issuance of intermediate certificates not only complicates the certification path, but also poses potential security risks. Durumeric et al. conducted an analysis of the HTTPS ecosystem and unexpectedly found a lot of cross-certificates among CA certificates which trusted by browsers in 2013 [7]. Roosa et al. believed that as CAs often do not disclose their cross-sign relationships when request to include in the root store, which exacerbates the opacity of the PKI system [22]. Hiller et al. systematically detailed the use and effect of cross-signing on the current web PKI, but did not analyze the behavior of cross-certificate issuance and discuss how to control it [10]. Casola et al. proposed an automatic methodology to enable CAs evaluate and compare the certificate policies for cross-certification [2]. However, most of above works have an emphasis on the measurement and analysis, but do not concentrate on the trust propagation control under the certification path, especially the trust originated from cross-certificates.

Based on the idea of transparency, several designs were proposed to improve the security and/or performance. A number of studies have reworked the log server structure to achieve more efficient transparency [23,24,29] and support more types of transparency [23,28], such as support for revocation transparency. PoliCert [27] records subject certificate policies and certificates in public logs, providing the cryptographic proofs of presence and absence. CONIKS [20] builds transparent key directories based on Merkle prefix trees, allowing users to audit their public keys while maintaining privacy. CTng [15] modifies the current CT design in a limited way to achieve transparency including certificate and revocation status without requiring any trusted third party.

## 6 Conclusion

In this paper, we analyze the problem of trust propagation with cross-certificates. We detail that the existing extension solution can only provide coarse-grained control over certificates. To achieve fine-grained control in cross-certificate trust propagation, inspired by the transparency in CT, we introduce the X-FTPC with

a new certificate extension to force the Cross-signed CA to submit the end-entity certificate to the specified log for pre-verification before it can be finally issued. X-FTPC realizes the real-time monitoring of the certificate issuing behavior of the Cross-signed CA, and achieves the goal of fine-grained control of the trust propagation over cross-certificate. Our research and analysis show that X-FTPC is effective in a variety of scenarios. In addition, X-FTPC has good compatibility with the CT mechanism widely deployed at present, thus enabling incremental deployment.

## References


1. Amann, J., Gasser, O., et al.: Mission accomplished? HTTPS security after DigiNotar. In: 17th IMC (2017)
2. Casola, V., Mazzeo, A., Mazzocca, N., Rak, M.: An innovative policy-based cross certification methodology for public key infrastructures. In: Chadwick, D., Zhao, G. (eds.) EuroPKI 2005. LNCS, vol. 3545, pp. 100–117. Springer, Heidelberg (2005). https://doi.org/10.1007/11533733_7
3. Chung, T., Liu, Y., et al.: Measuring and applying invalid SSL certificates: the silent majority. In: 16th IMC (2016)
4. Clark, J., van Oorschot, P.: SSL and HTTPS: revisiting past challenges and evaluating certificate trust model enhancements. In: 34th IEEE S&P (2013)
5. Cooper, D., Santesson, S., et al.: IETF RFC 5280 - Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile (2008)
6. Debnath, J., Chau, S.Y., et al.: On re-engineering the X.509 PKI with executable specification for better implementation guarantees. In: 28th ACM CCS (2021)
7. Durumeric, Z., Kasten, J., et al.: Analysis of the https certificate ecosystem. In: 13th IMC (2013)
8. Google Inc.: Certificate transparency (2021). https://www.certificate-transparency.org/
9. Google Inc.: Known logs (2021). https://www.certificate-transparency.org/known-logs
10. Hiller, J., Amann, J., et al.: The boon and bane of cross-signing: shedding light on a common practice in public key infrastructures. In: 27th ACM CCS (2020)
11. Holz, R., Braun, L., et al.: The SSL landscape: a thorough analysis of the X.509 PKI using active and passive measurements. In: 11th IMC (2011)
12. Internet Security Research Group: Chain of Trust (2021). https://letsencrypt.org/certificates/
13. Johnathan Nightingale: Mozilla Security Blog - DigiNotar Removal Follow Up (2011). https://blog.mozilla.org/security/2011/09/02/diginotar-removal-follow-up/
14. Laurie, B., Langley, A., et al.: IETF RFC 6962 - Certificate transparency (2013)
15. Leibowitz, H., Ghalwash, H., et al.: CTng: secure certificate and revocation transparency. Cryptology ePrint Archive (2021)
16. Li, B., Lin, J., et al.: Certificate transparency in the wild: exploring the reliability of monitors. In: 26th AMC CCS (2019)
17. Li, B., Lin, J., et al.: Locally-centralized certificate validation and its application in desktop virtualization systems. IEEE TIFS **16**, 1380–1395 (2020)
18. Li, B., Lin, J., et al.: The invisible side of certificate transparency: exploring the reliability of monitors in the wild. IEEE/ACM ToN **30**(2), 749–765 (2021)

19. Matsumoto, S., Szalachowski, P., Perrig, A.: Deployment challenges in log-based PKI enhancements. In: 8th EuroSec (2015)
20. Melara, M.S., Blankstein, A., et al.: CONIKS: bringing key transparency to end users. In: 24th USENIX Security Symposium (2015)
21. Mozilla: Bug 403437 - Request Valicert/Starfield/GoDaddy Root Certificates be enabled for EV. https://bugzilla.mozilla.org/show_bug.cgi?id=403437
22. Roosa, S.B., Schultze, S.: Trust darknet: control and compromise in the internet's certificate authority model. IEEE Internet Comput. **17**(3), 18–25 (2013)
23. Ryan, M.D.: Enhanced certificate transparency and end-to-end encrypted mail. In: 21st NDSS (2014)
24. Singh, A., Sengupta, B., Ruj, S.: Certificate transparency with enhancements and short proofs. In: Pieprzyk, J., Suriadi, S. (eds.) ACISP 2017. LNCS, vol. 10343, pp. 381–389. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59870-3_22
25. StackExchange: Are X.509 nameConstraints on certificates supported on OS X? https://security.stackexchange.com/questions/95600/are-x-509-nameconstraints-on-certificates-supported-on-os-x
26. Stark, E., Sleevi, R., et al.: Does certificate transparency break the web? Measuring adoption and error rate. In: 40th IEEE S&P (2019)
27. Szalachowski, P., Matsumoto, S., et al.: PoliCert: secure and flexible TLS certificate management. In: 21st ACM CCS (2014)
28. Szalachowski, P., Chuat, L., et al.: PKI safety net (PKISN): addressing the too-big-to-be-revoked problem of the TLS ecosystem. In: 1st IEEE EuroS&P (2016)
29. Tomescu, A., Bhupatiraju, V., et al.: Transparency logs via append-only authenticated dictionaries. In: 26th ACM CCS (2019)
30. Turnbull, J.: Cross-certification and PKI policy networking. Entrust, Inc. (2000)
31. Zhang, Y., Liu, B., et al.: Rusted anchors: a national client-side view of hidden root CAs in the web PKI ecosystem. In: 28th ACM CCS (2021)