Certificate Cothority: Towards Trustworthy Collective CAs

Ewa Syta, Iulia Tamas, Dylan Visher, David Isaac Wolinsky, and Bryan Ford
Yale University

1. INTRODUCTION

Our online infrastructure depends on *authorities* that provide conceptually simple but security-critical services, such as timestamping, logging, directory listings, digital certificates, or randomness. Unfortunately, while we expect and rely on these authorities' trustworthiness, they might in fact be arbitrarily dishonest. Specifically, the current state of Certificate Authorities (CAs), a vital component to the public key infrastructure (PKI), exemplifies the far-reaching consequences of our dependence on organizations to make unilateral yet security-critical decisions.

Certificate Authorities sign certificates attesting that the holder of a public key legitimately represents a name such as `google.com`, to authenticate SSL/TLS connections. Only if a server can produce a certificate signed by a trusted CA, will the client's browser accept it and establish a secure connection. Current web browsers directly trust dozens of root CAs and indirectly trust hundreds of intermediate CAs, any one of which can issue fake certificates for any domain since only contractual obligations limit the scope of each CA. An adversary can easily use a fake certificate for attacks such as website spoofing and man-in-the-middle attacks.

Issues with CAs range from malicious actions, incompetency, cooperation with governments, and security breaches. It takes only one such CA to threaten the security of the entire PKI and in turn, everyone on the Internet. Due to this "weakest-link" security, hackers have stolen the "master keys" of CAs such as DigiNotar and Comodo issuing fake certificates covering a bulk of the Internet traffic. Moreover, CAs themselves abused certificate-issuance mechanisms as in the recent CNNIC/MCS incident, where the Chinese CA issued an unconstrained certificate allowing the Egyptian company MCS Holdings to produce certificates trusted by any major browser or operating system.

There is virtually no oversight over CAs beyond voluntary, industry organizations. Browser and OS vendors hold the most governing power over CAs as they can remove a CA from their root store in case of a misbehavior or compromise. This is not a reliable mechanism, however, after the recent breach Mozilla and Google removed CNNIC them from their root stores while Apple did not.

We now find ourselves in a situation where much of our sensitive communication happens on the Internet. This communication is only as secure as the weakest link in the CA system, with almost the only line of defense being a handful of browser and OS vendors who themselves do not have a unified vision of how to handle trust and security breaches. With applaudable efforts on the rise, such as "Let's Encrypt"[1] aiming to make encryption on the Internet ubiquitous, it is more important than ever to finally provide a solid foundation to the PKI system.

2. CURRENT DEFENSES

After many CA breaches, as a stopgap, browsers such as Chrome and Firefox hard-code *pinned* certificates for particular sites or particular CAs for each site – but browsers cannot ship with hard-coded certificates or CAs for each domain for the whole Web. Alternatively, browsers pin the first certificate a client sees protecting a site's regular users but not new users.

More general and currently pushed for mitigations for CA weaknesses rely on logging and monitoring certificates as proposed in systems like AKI [4], PoliCert [8], and the most popular and actually deployed Certificate Transparency [5].

Certificate Transparency requires CAs to insert newly-signed certificates into public logs, which a larger body of *monitors* and *auditors* check for consistency and invalid certificates. Monitoring can unfortunately detect misbehavior only *retroactively* – e.g., after a properly-signed but fake certificate has appeared – placing victims in a race with the attacker. Web browsers could check all certificates they receive against such logs and/or via multiple Internet paths but such checks add delays to the critical page-loading path. Further, these approaches assume Web users can connect to independent logging, monitoring, or relaying services without interference, but this assumption fails when the user's own ISP is the adversary – a scenario that has unfortunately become all too realistic whether motivated by state-level repression or profit.

3. OVERVIEW AND BUILDING BLOCKS

We propose to replace current, high-value certificate authorities with a *certificate cothority* (CC) – a practical system, which embodies strongest-link security by allowing all participants to validate certificates *before* they are issued and endorsed, and therefore proactively prevent their misuse.

We build certificate cothorities using an instantiation of a large-scale, general *collective authority*, which we call *cothority*. The basic goal is to split trust across a large and diverse body of independently-run servers. Each of potentially thousands of hosts comprising a cothority independently validates each public output, contributing a share of a *collective* digital signature to each validated output, or withholding its signature and raising an alarm if misbehavior is detected. Clients can validate a cothority's output – such as an endorsed certificate, but also timestamp, or random number – using a single inexpensive cryptographic operation comparable to conventional signature verification. This collective signature attests to the client that not just one but *many* well-known servers (ideally thousands) independently checked and signed off on that output. Therefore, a cothority guarantees *strongest-link* security whose strength increases as the collective grows, instead of decreasing to weakest-link security as in today's CA system.

Below we briefly summarize our ideas on how to build general cothorities but refer the reader to our technical report [7] for details.

CoSi: Collective Signing

Each cothority is implemented by a single instance of CoSi, the first practical *collective signing* protocol we know of to build large-scale cothorities based on existing schemes such as threshold signatures, aggregate signatures and multisignatures [1, 6],

First, to allow CoSi to scale to thousands of servers, we limit the computation and network bandwidth costs imposed on each participating server by using tree-based communication structures comparable to those long used in multicast protocols [2].

[1] `https://letsencrypt.org/`

In a single round of CoSi, the leader, a distinguished node trusted only with availability, proposes a new record – a single element or a cryptographic summary of many elements in a form of a Merkle tree built on those elements – to be validated and publicly recorded in a tamper-evident log. Then, the leader coordinates with all of its signers to validate the log entry and generate a collective signature for it, which we implement using a Schnorr multisignature [1, 6]. The result is a collectively signed log entry, which anyone may check against the cothority roster.

Schnorr multisignatures are especially suitable for cothorties: *(i)* each signer retains its own private/public key pair and the constant-size collective signature verifies under a single, aggregate public key efficiently calculated using only individual public keys; and *(ii)* the signature can be *incrementally* generated in a leaf-to-root traversal of a communication tree in a way that each server's computation and communication costs depend directly only on its number of immediate children, yielding $O(\log N)$ per-node costs in a $O(\log N)$-depth tree.

We have built a working cothority server prototype[2] implementing collectively signed logging, timestamping, and vote-counting services. Experimental evaluations on DeterLab [3] demonstrate that the prototype scales easily to over 4000 participant servers, handles hundreds of thousands of client requests per second in aggregate, with typical latencies of only 1–5 seconds – delays easily tolerable by typical authority services.

4. CERTIFICATE COTHORITY

Our goal is to demonstrate that replacing the current "weakest-link" CA system with a "strongest-link" certificate cothority is compelling and practical. The encouraging performance results of our general cothority prototype lead us to believe that we can build a trustworthy, collective CA system using theoretically established and well-understood techniques.

Principles of Operation

We envision federating today's hundreds of CAs along with browser vendors and security companies into a single certificate cothority. In our architecture, each participant can validate certificates proposed by all other CAs *before* they are collectively signed, raising an alarm and *proactively* preventing the signing of fake certificates in the first place. Once signed, each certificate would still be verified by a single signature, but that signature would embody much stronger and broader-based *trust* of the entire certificate cothority.

A certificate cothority periodically, depending on the volume of certificate requests, invokes a new certificate signing round to collectively endorse newly-generated certificates. During each round, each participating CA proposes a set of new certificates, which is made available for everyone to verify. Hence, every CA is given an opportunity to inspect all newly-proposed certificates and to watch for and proactively block the signing of unauthorized certificates, such as certificates proposed by a CA that is not recorded as having contractual authority over a given domain. There are several ways for CAs to validate certificates. For example, the CA currently responsible for a domain such as `google.com` could verify that no other CA proposes a `google.com` certificate. Alternatively, each CA could use a *global CC policy* defining the scope of each CA's authority and perhaps the minimal security and validity requirements (*e.g.*, currently recommended key lengths or encryption algorithms) for certificates.

After the *collective verification* phase, the CC moves to the *collective approval* phase. In this phase, each CA is given an opportunity to raise objections to any improper certificates, implicitly accepting the remaining certificates. In the *collective signing phase*, CC employs CoSi to only sign certificates without any objections. CC may utilize administrative oversight to deal with certificates flagged as bad. Alternatively, CC could use a voting-based override (also easily accomplished using cothorities) if the majority of CAs find the certificates in questions to be valid or if a given CA is being deliberately malicious by trying to stall the progress of CC.

[2] `https://github.com/DeDiS/prifi/tree/master/coco`

Deploying a Certificate Cothority

While deploying a certificate cothority would be challenging, our approach offers incremental deployment options with backward compatibility for the existing infrastructure. We envision three possible incremental deployment models. In a *browser-centric* certificate cothority, the browser vendor can act as a leader of the cothority, encouraging (and after a sufficient transition period, demanding) that root and subsidiary CAs currently included in the browser's root store join the browser's cothority, if they wish to remain included. In a *root-CA-centric* certificate cothority, the root CA may decide to transition its master key gradually to a collectively signed key and encourage (and again, eventually require) its subsidiary delegated CAs to join its cothority instead of wielding independent delegated CA power. In a *CT-centric* certificate cothority, a collection of independent Certificate Transparency log, monitor, and/or auditor servers form a cothority and collectively endorse CT-style signed certificate timestamps (SCTs). Those SCTs can be then included into existing certificates using X.509v3 extension fields, without otherwise requiring any changes to the actual certificate signing algorithms or legacy browsers. Web browsers of course would need to be gradually upgraded to support Schnorr signatures. During their transition period root CAs could retain traditional root CA keys for use in older web browsers while embedding a certificate cothority key instead into suitably upgraded browsers.

We envision that the demand by users, companies, and browser and OS vendors alike for increased security would greatly assist in encouraging CAs' early participation and investment in a CC infrastructure. We take the deployment of Certificate Transparency as an encouraging sign of changes to come and feel that there is no technical reason to continue with the centralized, weakest-link security we seem to have settled for.

5. REFERENCES

[1] M. Bellare et al. Multi-signatures in the plain public-key model and a general forking lemma. In *CCS*, 2006.

[2] M. Castro, et al. SplitStream: high-bandwidth multicast in cooperative environments. In *SOSP*, 2003.

[3] DeterLab network security testbed, September 2012. `http://isi.deterlab.net/`.

[4] T. H.-J. Kim, et al. Accountable key infrastructure (AKI): A proposal for a public-key validation infrastructure. In *WWW*, 2014.

[5] B. Laurie, et al. Certificate transparency, June 2013. RFC 6962.

[6] S. Micali, et al. Accountable-subgroup multisignatures. In *CCS*, 2001.

[7] E. Syta, et al. Decentralizing authorities into scalable strongest-link cothorities. *arXiv preprint arXiv:1503.08768*, 2015.

[8] P. Szalachowski, et al. PoliCert: Secure and flexible TLS certificate management. In *CCS*, 2014.