Certificate Transparency

CS5321 (Network Security): Week 4 Paper Summary Presentation

Daniel W. K. Loo

Certificate Transparency [Laurie 2014]

Overview

- Trusted 3rd party, certificate authorities (CAs) can be compromised (e.g. DigiNotar, MonPass)

- "State of the art in software engineering and system design cannot render anyone absolutely safe"

- Problem: entity authentication (is the user communicating with the domain owner of the URL?)

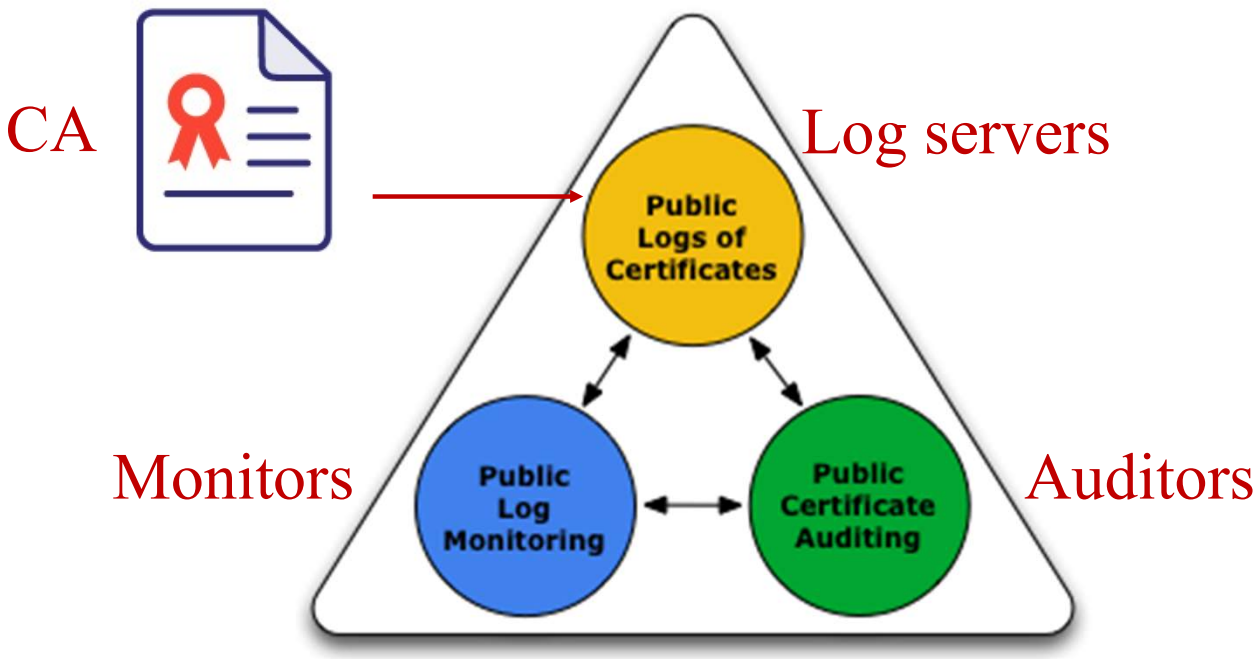
Threat and problems

- Attackers compromise CAs, inserting their own certificates
- Malicious CA intentionally issuing fraudulent certificates
- Broken chain of trust
- Issues with other suggested solutions:
  - Pinning: Not scalable
  - Notaries: Window between scan updates, attacker with wide distribution
  - DNSSEC: Add another trusted party on top?

Author's proposed solution

- Certificate Transparency (CT), public, verifiable, append-only log
- Open framework for auditing certificates
- Anyone who wants to issue a certificate can log it (publicly viewable) and interested parties can take action if it is invalid
- Merkle trees hold hashes of the logs to quickly verify:
  - Is the certificate in the log?
  - Has it been tampered with?

Author's proposed solution



Trade-offs

- Uses signed certificate timestamps (SCT) and maximum merge delay (MMD)
- Require storage redundancy once SCT issued
- MMD needs to be long enough to permit a steady-state to be reached
- Multiple logs for certificates that have longer lifetimes
- Amount of information to include in the log

Concerns

- Merkle Hash Trees (MHT): Efficient “proof mechanism”?
- Append only log: Storage?
- Synchronisation between CT servers?
- Latency: Certificate registration → Proof of Inclusion
- Privacy Issues: Private domains also registered
- Detection? Yes, but what about revocation?

Efficient “proof mechanism”

- CT server supports geographical area (performance)
- Base MHT provides proof of membership, but not direct proof of non-membership
- Sorted Merkle trees more efficient for that but append (insert) function more complex
- Consider exploring other structures: strong accumulators[1], sparse Merkle trees[2], RSA accumulators[3], or bilinear map accumulators[4]

One-way accumulators: a decentralised alternative to digital signatures [Benaloh & de Mare ‘94]

“Proof mechanism” references

[1] Philippe Camacho, Alejandro Hevia, Marcos Kiwi, Roberto Opazo, "Strong Accumulators from Collision-Resistant Hashing", International Conference on Information Security, ISC 2008: Information Security, pp 471–486, 2008

[2] Rasmus Dahlberg, Tobias Pulls, Roel Peeters, "Efficient Sparse Merkle Trees: Caching Strategies and Secure (Non-)Membership Proofs", NordSec 2016, 2016

[3] Jan Camenisch, Anna Lysyanskaya, "Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials", CRYPTO 2002: Advances in Cryptology, pp 61–76, 2002

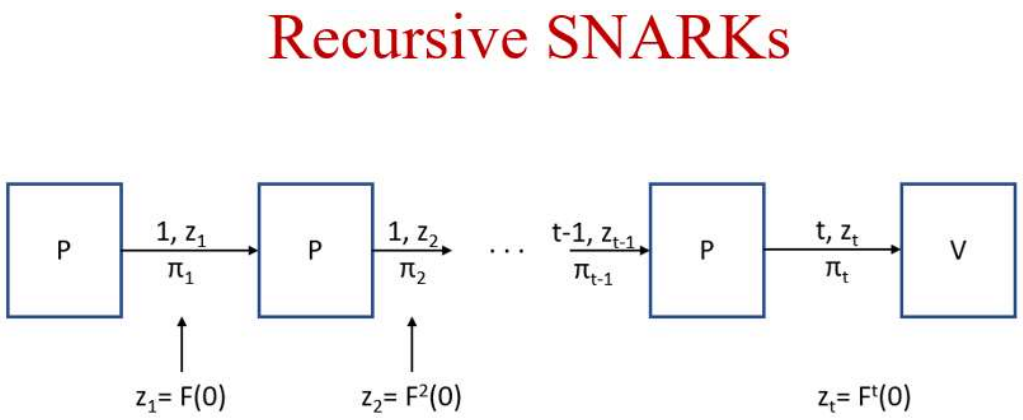
[4] Lan Nguyen, "Accumulators from Bilinear Pairings and Applications to ID-based Ring Signatures and Group Membership Revocation", CT-RSA 2005: Topics in Cryptology (CT-RSA 2005), pp 275–292, 2005

Append-only logs: Storage

- Storage requirements grow over time
- Recursion? Recursion!

Single log per source as recursive proof

- Current proof/log also proves previous log is valid, and so forth
- Only the most recent log/proof required
- Not a new idea, also used in blockchain (ZK-SNARKs)



Nir Bitansky, Ran Canetti, Alessandro Chiesa, Eran Tromer, "Recursive Composition and Bootstrapping for SNARKs and Proof-Carrying Data", STOC '13: Proceedings of the forty-fifth annual ACM symposium on Theory of Computing, Pages 111–120, June 2013

Single log per source as recursive proof

- Append-only → append/update-only
- Trade-off: Reduced storage requirements vs possible attacks
- Even append-only vulnerable to “attacker only needs to win once”
  - Backdoor installed due to accepted certificate before revocation
  - Examples: NSA backdoor to elliptic curve cryptography (Dual_EC), DoublePulsar cryptojacker

Other issues to think about

- Synchronisation: Partitioning attack via BGP hijack. Which side to trust after reconnection? Perhaps a consensus mechanism…

- Latency

- Privacy

Thank you. Questions?