DTKI

A Protocol for Public-Key Validation

Jiangshan Yu, Vincent Cheval, and Mark Ryan

School of Computer Science
University of Birmingham

September 2013

Overview

1. Background
   - Public-key certificate issues
   - Existing proposals

2. Our work
   - Properties
   - Proposed protocol

1. Background
   - Public-key certificate issues
   - Existing proposals
2. Our work
   - Properties
   - Proposed protocol

Certificate

A certificate is a signed statement that bind a public key to a subjects details. (*Example*)

Certificate

A certificate is a signed statement that bind a public key to a subjects details. (*Example*)

Loosely speaking:

$$Cert_{Subj} = Sign_{SK_{Issuer}}(ID_{Subj}, PK_{Subj})$$

Web Certificate Verification

CA/B trust model

- browser defines a set of CAs;
- browser accepts all certificates issued by any one of them.

Mozilla Firefox browser initially trusts 57 root CAs.

The EFF SSL Observatory : $\sim$ 1500 of CAs in total.

Issues

Problems with CA/B

- Any CA can certify public keys for any domain.
  (Thus, we have to assume that all CAs are trustworthy.)
- CA/B has no efficient mechanism to detect mis-issued certificate.

Issues

Problems with CA/B

- Any CA can certify public keys for any domain.
  (Thus, we have to assume that all CAs are trustworthy.)
- CA/B has no efficient mechanism to detect mis-issued certificate.

Example of Attacks:

- Comodo was attacked and fake certificates were issued for popular domains (e.g. Google, Yahoo, Skype, etc.). (2011)

- DigiNotar issued 531 fake certificates for more than three hundred domains, including most of major Internet communications companies. (2011)

Does it matter?

Does it matter?

If you encrypt with the wrong key, the attacker may get your message.

Existing Proposals

Table : Taxonomy of existing solutions

Taxonomy	**Existing Proposals**
PGP adoption	MonkeySphere;
DNS extension	DANE
Difference observation	SSL Observatory; Certificate Patrol; Perspectives;
	DoubleCheck; CertLock; Covergence;
	TACK.
Public log adoption	Sovereign Keys; Certificate Transparency;
	AKI

Certificate transparency [Laurie, Kasper, Langley 2012]

Basic idea:

- All certificates issued by a CA should be recorded in a public log.
- To accept a certificate, browsers must verify the proof such that this certificate is included in the log.
- Domain owners can detect mis-issued certificates by checking the log.

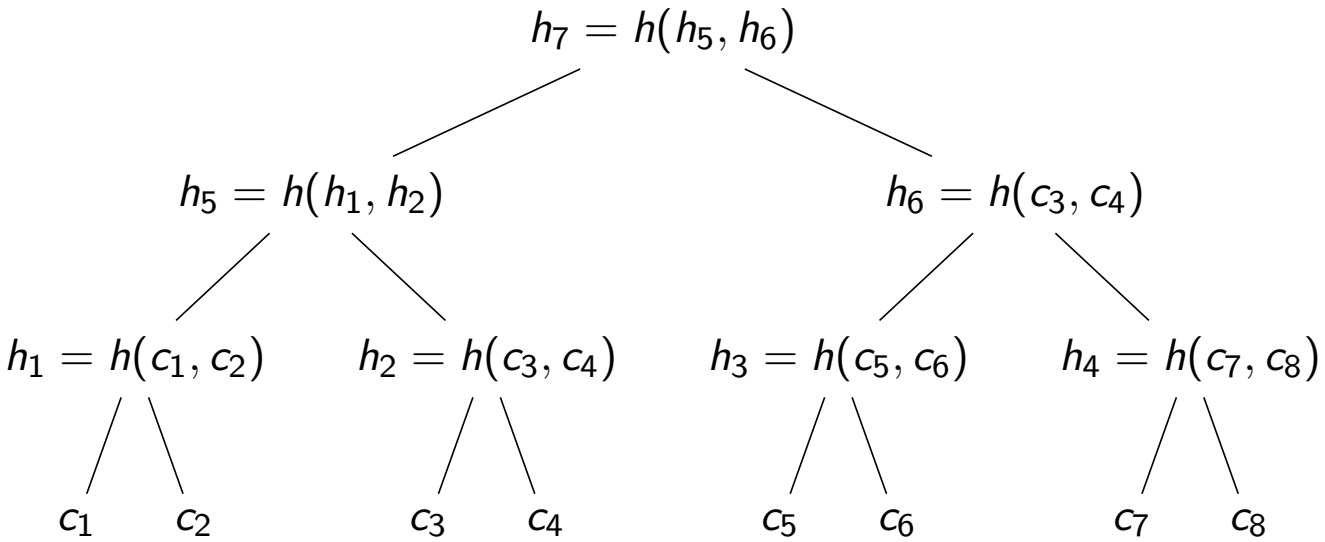
IETF RFC6962 (June 2013).

Public Log

Desired proofs:

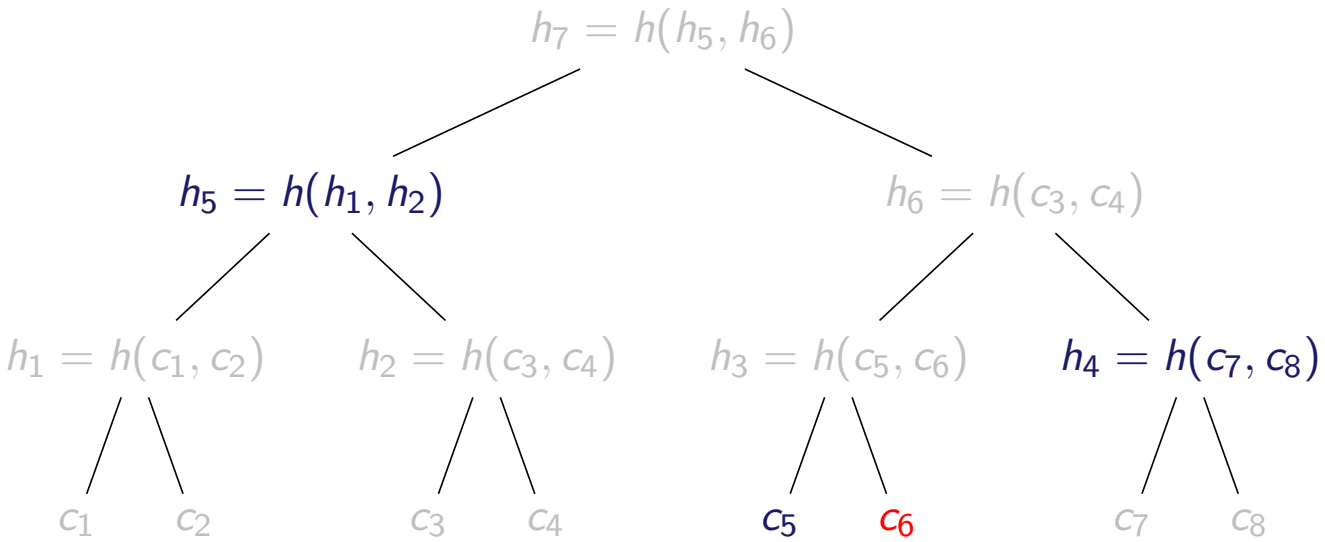
- **Proof of presence** proves that a certificate is included in a public log.
- **Proof of extension** proves that the current public log is an extension of previous versions.
- **Proof of currency** proves that the public key of a subject is the latest one in the public log.
- **proof of absence** proves that no certificate in the log is issued for the given subject.

Append-only public log – Merkle tree



Proof of	Complexity
presence	$O(log\, n)$
extension	$O(log\, n)$
currency	$O(n)$
absence	$O(n)$

Append-only public log – Merkle tree

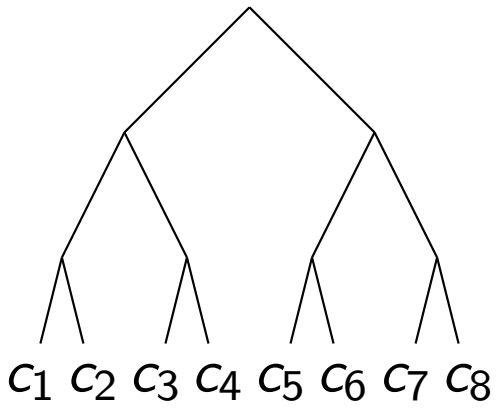


Proof of	Complexity
presence	$O(log\, n)$
extension	$O(log\, n)$
currency	$O(n)$
absence	$O(n)$

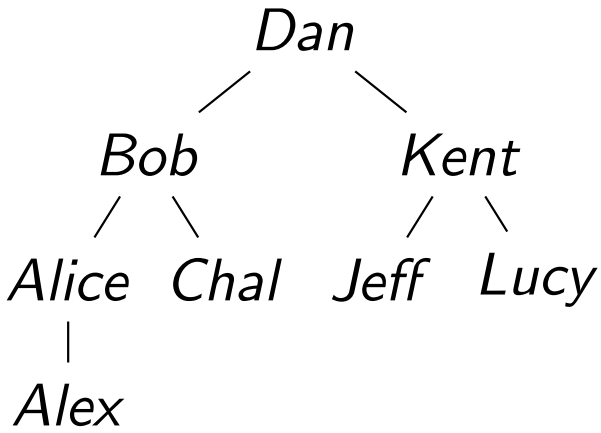
An improvement

Certificate Issuance and Revocation Transparency [Ryan 2013]

ChronTree



LexTree



Proof of		
presence	$O(\log n)$	$O(\log n)$
extension	$O(\log n)$	$O(n)$
currency	$O(n)$	$O(\log n)$
absence	$O(n)$	$O(\log n)$
consistency	$O(n)$	

Consistency Proof

- Public auditor.
- Random checking by clients.

Problems

Difficulty with absence proof

CT allows multiple public logs.

- **Efficiency**: to verify currency proof and absence proof, a client/server needs to check all existing logs.
- **Security**: hide party attacks could be launched.

1. Background
   - Public-key certificate issues
   - Existing proposals

2. Our work
   - Properties
   - Proposed protocol

Basic properties

- In-band verification
- Built-in key revocation
- Ability to limit trust scope
- Domain ownership change protection
- Scalability
- Usability

Novel properties

Country neutrality

- Infrastructure providers (e.g. root CAs, timeline servers, and log providers) should not be dominated by a single country.
  - e.g. An Irish citizen accessing an Irish bank should be able to use Irish infrastructure.
- Government agencies who have compelled authorities in their country should not be able to use fake certificates without being readily detected.

Novel properties

Canonical signer

- For a given domain, anyone should easily identify parties that are authorised to establish key authenticity.

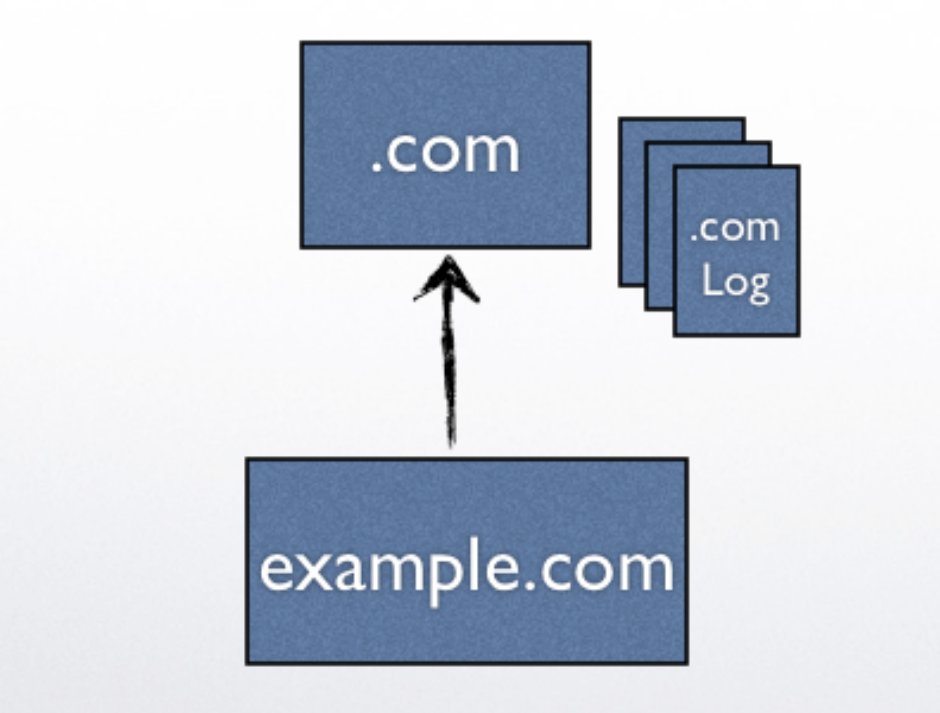
No-monopoly

- No provider should have a uniquely privileged position.
- (Trust agility) Any entity can freely remove his/her trust anchor without affecting the function of internet services.

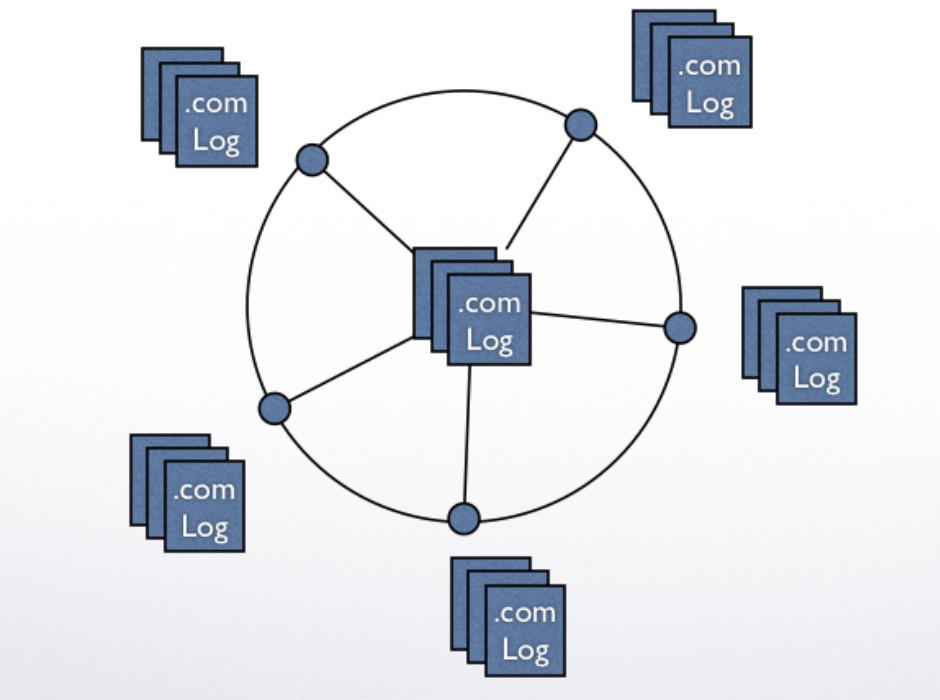
Resistance to hidden party attack

- "Hidden party" attack:
  A party issues certificates/logs/... for a subject that does not know that the party exists.
  E.g. CT log provider of a small organisation.

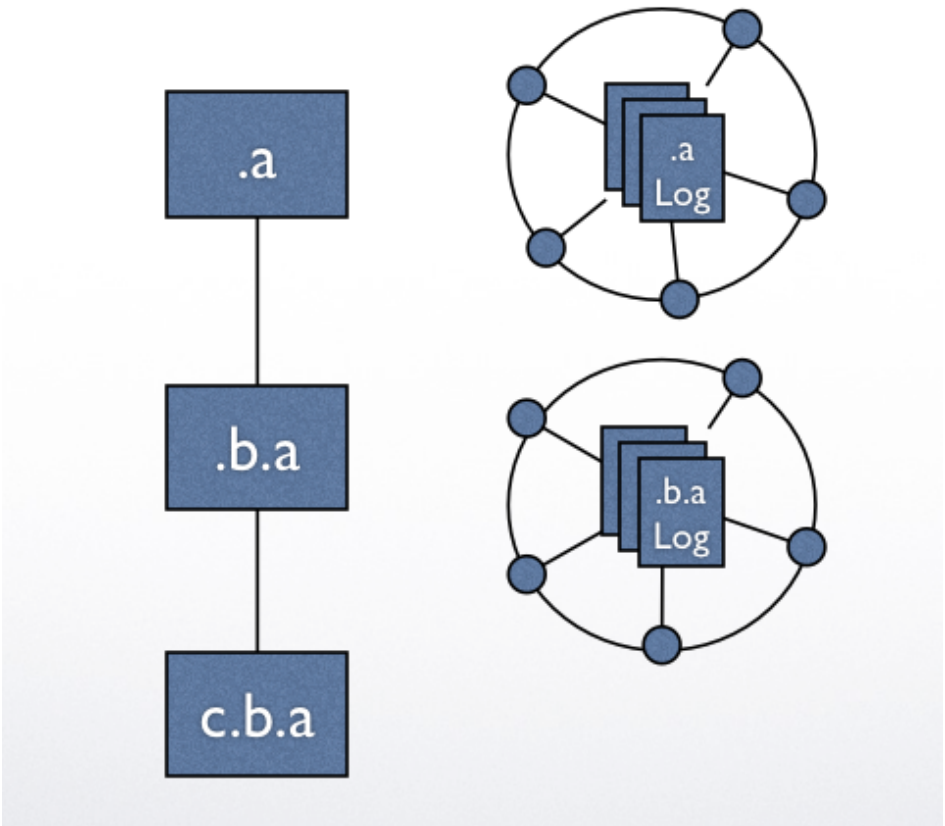
DNS-based Transparent Key Infrastructure (DTKI)



DNS-based Transparent Key Infrastructure (DTKI)



DNS-based Transparent Key Infrastructure (DTKI)



DTKI

DNS-based Transparent Key Infrastructure (DTKI)

Basic idea

- Parent domains certify public keys for their child domains;
- Parent domains should maintain a public log to record their child domains' certificates.
  - Each top level domain (*e.g.* .com, .net, and .uk) must maintain a log;
  - each second level domain (*e.g.* .co.uk and example.com) may or may not maintain a log;
    e.g. .co.uk does need to maintain a log, but example.com does not.
- Each log should be distributed to many peers which could be any interested party.

DTKI

The format of the log:

The log is organised as a ChronTree and a LexTree.

$$Log_{.com} = [(a.com, EK_{a.com}, VK_{a.com}), ...]$$

On-going work

- Proof transmission:
  - users query proofs over DNS (similar as DANE); or
  - users query proofs from mirrors in parallel with ServerHello phase, or
  - users cache the certificate obtained from TLS handshake, and check proofs with mirrors later.
- Formal proof.

Thank You!