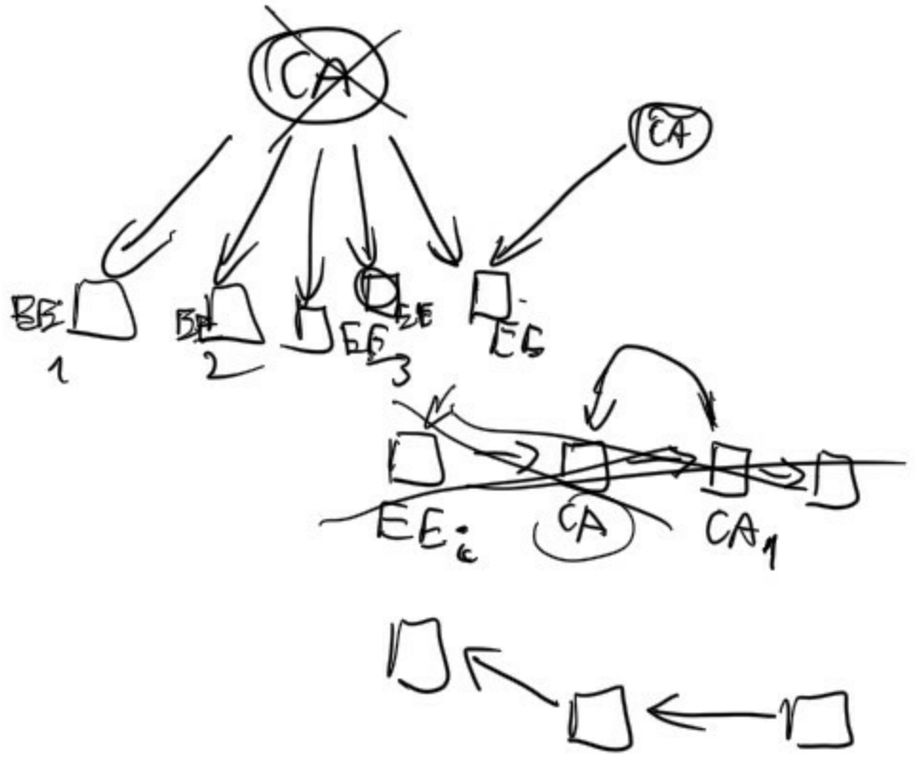
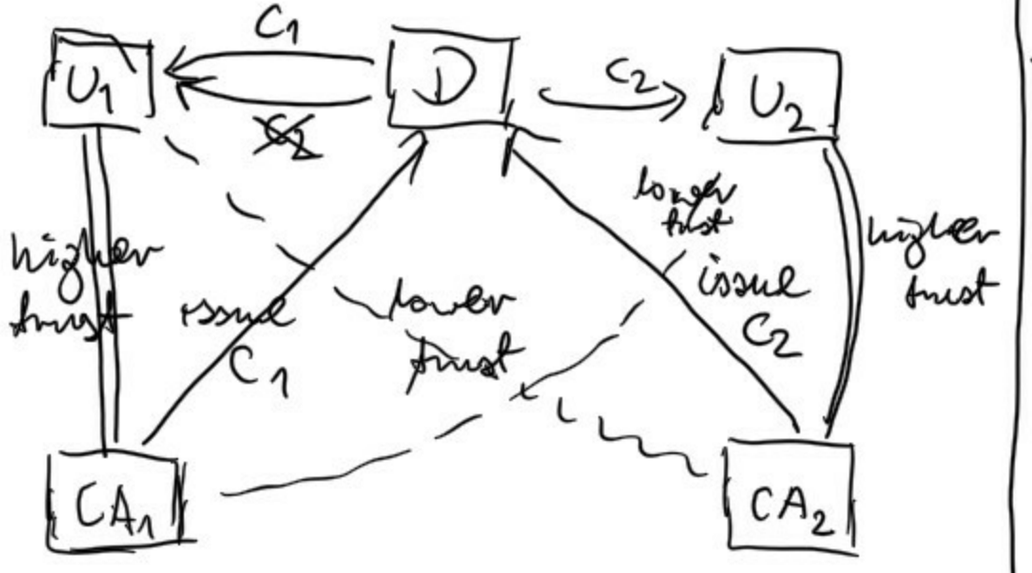


User U_1 trusts CA_1 more than CA_2 for issuing certificates for domain D because CA_1 supports multiperspective domain validation.

U_2 trusts CA_2 more than CA_1 because CA_2 is American CA and D's top-level domain (TLD) is US.



The idea in the paper is:

- each domain can set a domain policy
- each client (browser/verifier) can set a validation policy using trust levels for CAs.

Domain owners can specify policies in their certificates (it is important to make sure that the author of the policy is the owner of the private key corresponding to the public key present in the certificate).

But what if a malicious CA will issue a certificate for the domain, without any policy?

The idea is that for a given domain the client will receive all the certificates issued for that domain at last by the trusted CAs.

F-PKI allows the client to express preferences to certain CAs for a given name.

So the notion of trust is Ternary and name dependent: every authority may be either:

- untrusted,
- trusted,
- highly trusted.

In our example:

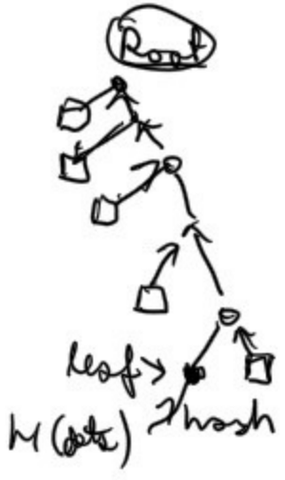
$U_1 \rightarrow CA_1$ highly trusted

CA_2 is trusted for domain D.

So the policy must be included in a certificate extension.

To not overlook (or to be sure to receive all) certificates and revocations for a given domain there is a service providing the data together with the verifiable proof that all entries are given.

The server is the extension of the Certificate Transparency Framework.



The server is also able to prove that some leaf is absent, and the proof is verifiable!

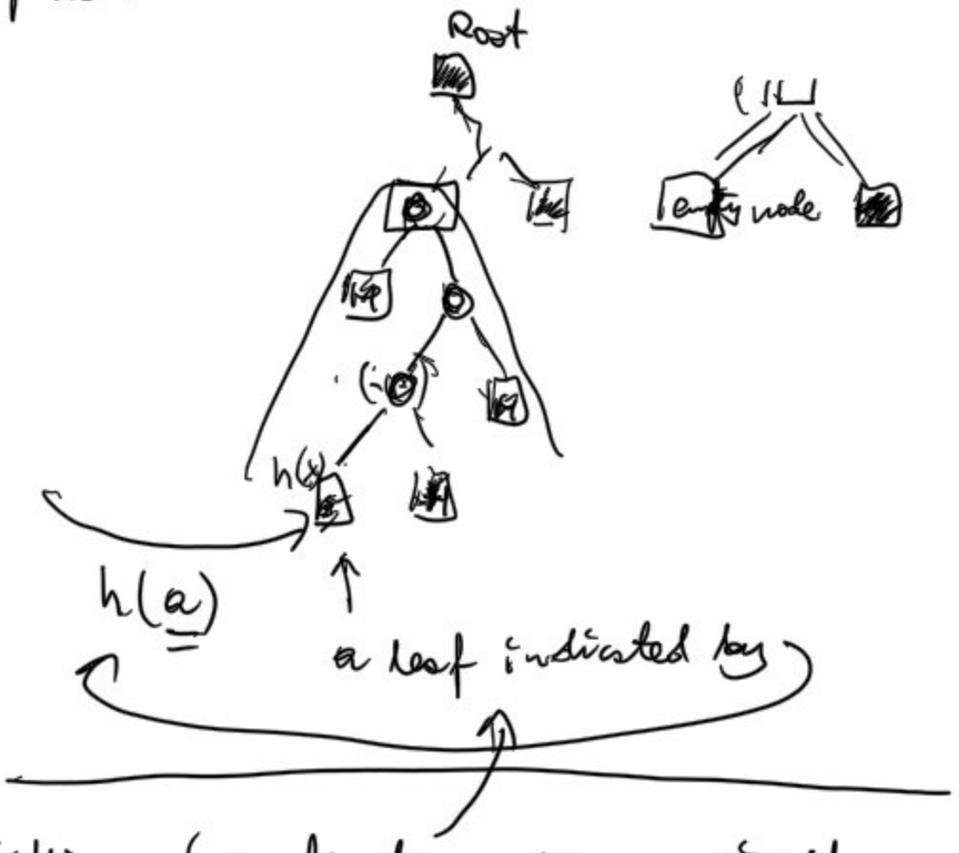
- sparse Merkle Trees are used

We gain a second level of assurance:

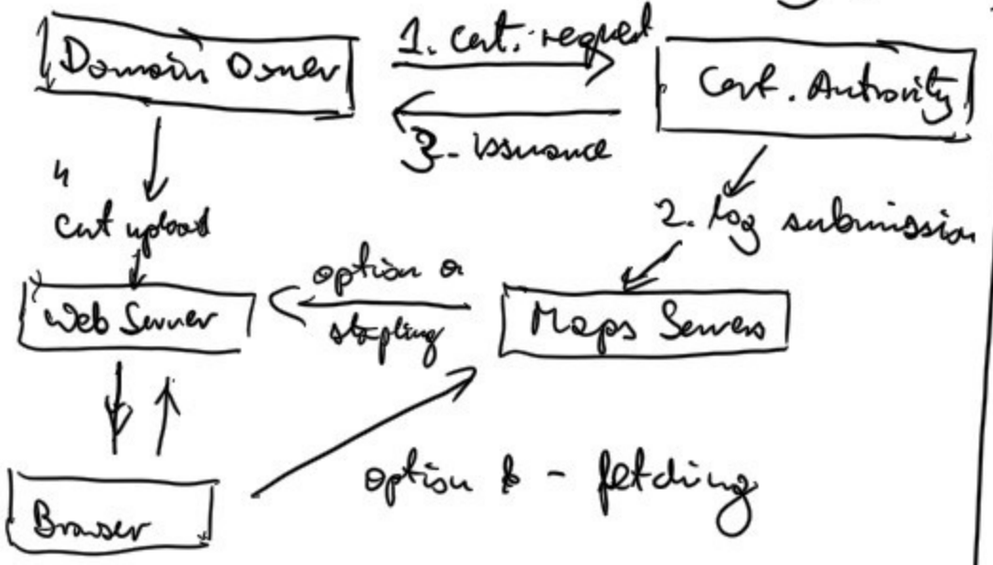
> not just verification that the certificates on a locally presented chain are not revoked and the chain is consistent

but having these services we are able to check more globally (CT or F-PKI) if there are some malicious CA producing alternative path.

Sparse-Merkle-tree:



Trillian (in that or in a similar way it works)



Domain policies:

ISSUERS (Set attribute): set of public keys (CAs) that may be used to verify D's certificate.

SUBDOMAINS (set attribute): set of subdomains names for which certificates can be issued (wildcards can be used).

example.com → *.shop.example.com

→ *.ftp.example.com

If no extension is present then all subdomain names are allowed.

WILDCARD_FORBIDDEN (Bool attribute): prohibits wildcards,

MAX_LIFETIME (Max_Attribute) - max certificate lifetime.

Domains are encouraged to obtain certificates from different CAs (for increased resilience to CA compromise) - if there is a difference (over time) in the policies the client takes the strictest one:

bool attributes: take and

max attributes: take min value,

set attributes: $S_1 \cap S_2$

Browser (client) policy:

CA-based - some CAs are more trusted (browser vendors evaluate CAs, security incidents etc.)

TLD-based: .us more trusted for US citizens,

in case of domains for *.gov.us

Organization Slices - domains owned by the company will probably obtain certificates from company's CA.

or a policy of bank.

The client policy may be downloaded by a client together with the browser or from the service (e.g. from bank). ← trust package.

So the resolution on the level of Domain Names is very important.

Certificate Validation Algorithm

↳ see Algorithm 1 (page 8) in the paper.