Something Something Certificate Transparency

Kiki der Gecko@Hackover 2015

What is this all about?

We all agree: SSL PKI is broken.

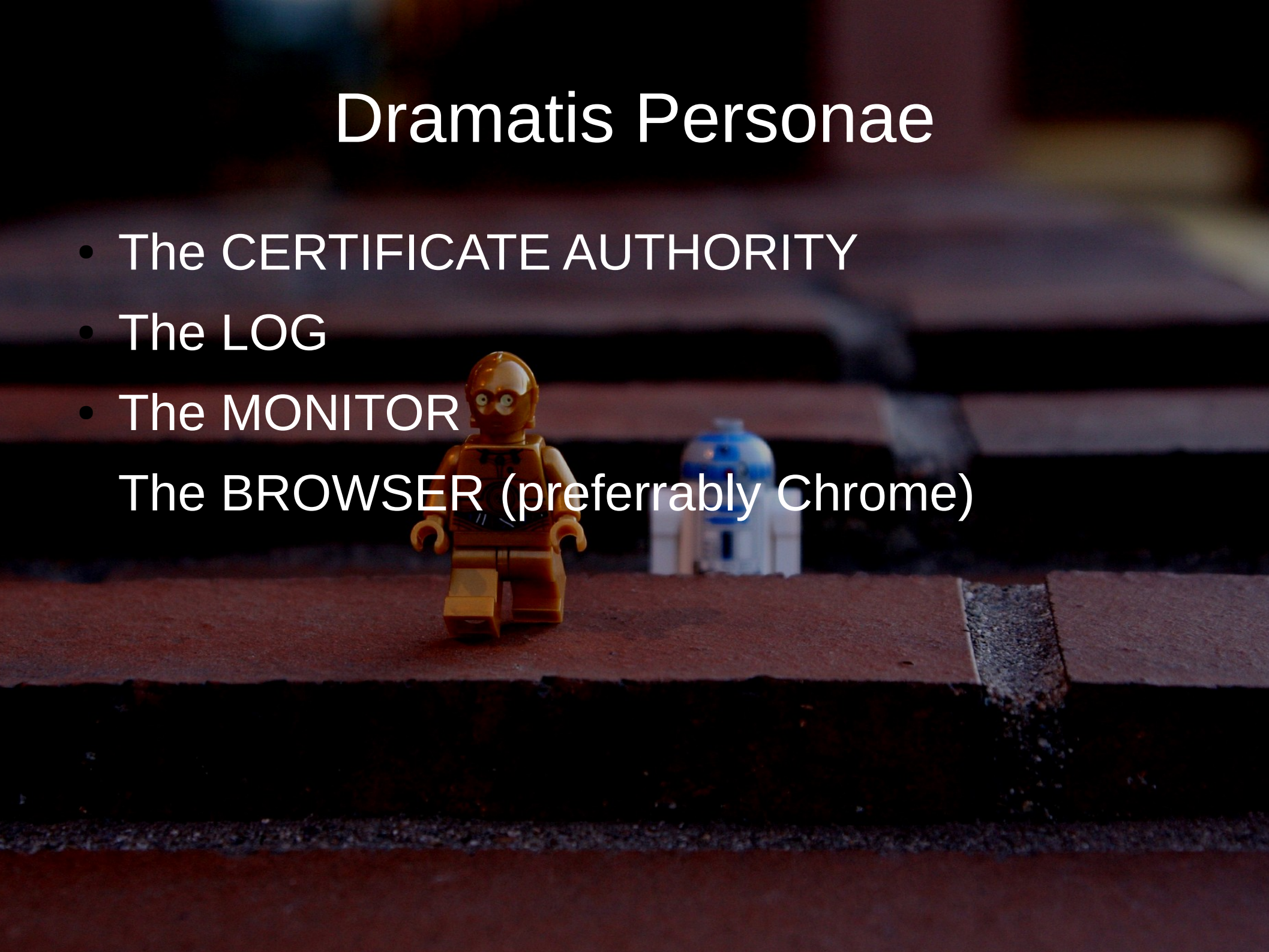
But don't fear: Some Google people (Laurie, Langley, Käsper) fixed this.

This is about RFC 6962.

Dramatis Personae

- The CERTIFICATE AUTHORITY
- The LOG
- The MONITOR

  The BROWSER (preferrably Chrome)



The Certificate Log

A log is a single, ever-growing, append-only Merkle Tree of [...] certificates.

The Certificate Log

A log is a single, ever-growing, append-only Merkle Tree of [...] certificates.

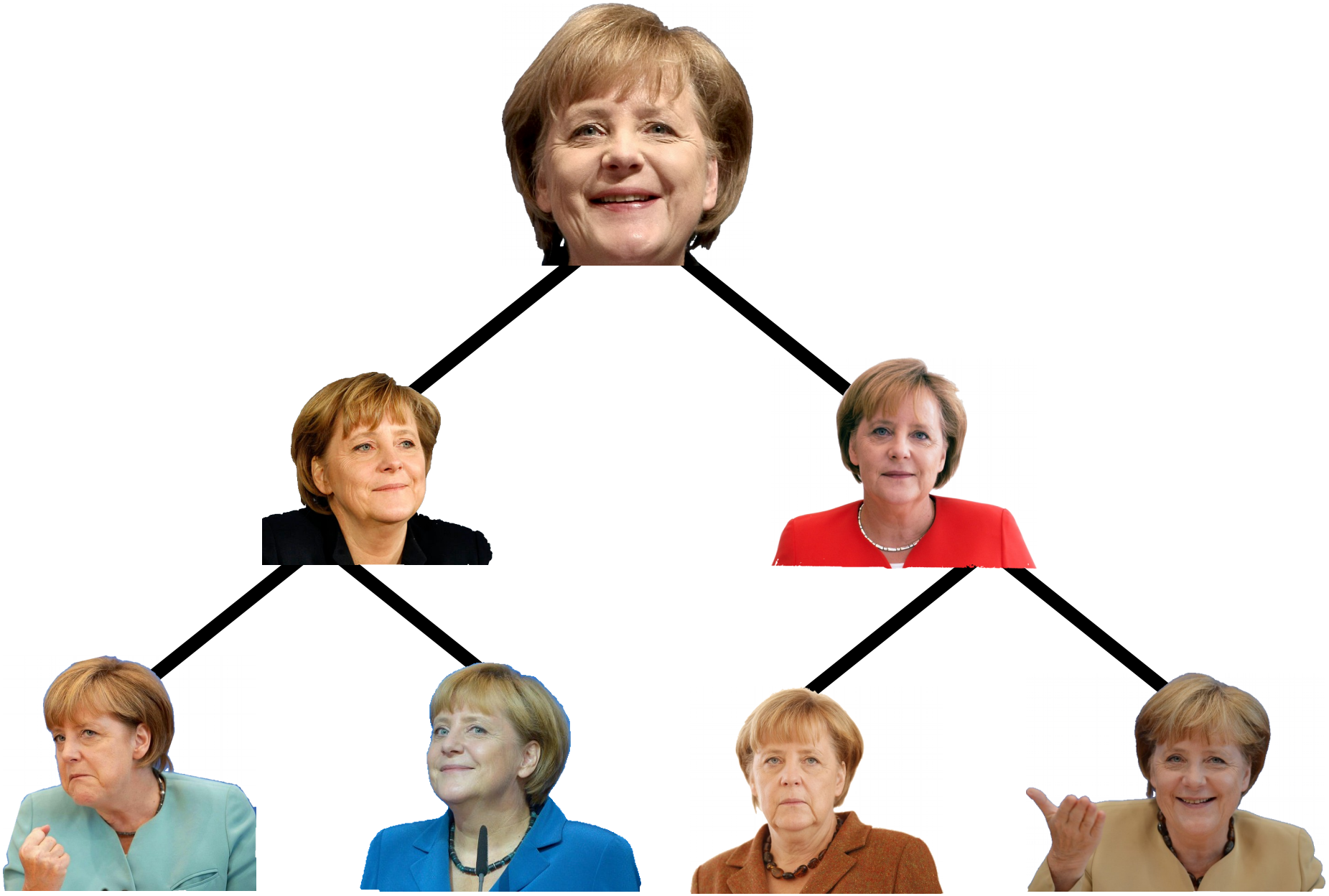
Merkle Tree

The Certificate Log

A log is a single, ever-growing, append-only Merkle Tree of [...] certificates.

Merkle Tree

Merkle




Fear Not! Introducing:
The Merkle Tree!

Why do we need this again?

- Public CAs will submit all their certificates to the untrusted, publicly auditable log.
- Logs can and should be monitored.
- Surprise: Merkle Trees happen to be append-only and easily auditable.

Tree Hashing 101

The hash of an empty list is the hash of an empty string:

MTH({}) = SHA-256().

- The hash of a list with one entry (aka leaf hash) is:

MTH({d(0)}) = SHA-256(0x00 || d(0)).

Tree Hashing 101

For n > 1, let k < n <= 2k. The Hash of an n-element list D[n] is defined recursively as

MTH(D[n]) = SHA-256(0x01 || MTH(D[0:k]) || MTH(D[k:n])),

where D[k1:k2] denotes the list {d(k1), d(k1+1),..., d(k2-1)} of length (k2 - k1).

(Note that the hash calculations for leaves and nodes differ. This is required to give second preimage resistance.)

Consistency Proofs

I said append-only, eh?

A consistency proof for MTH(D[n]) and a previously advertised hash MTH(D[0:m]) of the first m leaves (m <= n) is the list of nodes in the Merkle Tree required to verify that the first m inputs D[0:m] are equal in both trees.

Auditing

An audit path for a leaf is the shortest list of additional nodes in the Merkle Tree required to compute the Merkle Tree Hash for that tree.

If the root computed from the audit path matches the true root, then the audit path is proof that the leaf exists in the tree.


Onward, to the real world!

Building a Log

- We start with a list of accepted root CA certs
- Anyone can send certificates to a log (including the validation chain)

  → **Yep, no self-signed certificates in logs**
- The log returns a Signed Certificate Timestamp (SCT), the promise to include the log until a certain point in time.

Making TLS Love

- *TLS servers MUST present at least one SCT from one or more logs to the TLS client together with the certificate.*
- *TLS clients MUST reject certificates that do not have a valid SCT for the end-entity certificate.*
- More is better, since a client may mistrust specific logs.

Birds and Bees

The SCT must be included in the TLS handshake, by either:

- Using an X509v3 extension
- TLS extension “signed_certificate_timestamp“
- OSCP stapling

Clients must implement all mechanisms, servers at least one.

Let's Have a Threesome

- Log messages are sent as HTTP GET or POST requests
- Responses and POST parameters are encoded as JSON

In addition, clients should gossip about STHs.

Monitoring

For each log, do:

- Fetch current STH and verify signature
- Fetch all entries corresponding to the STH
- Confirm tree production from the fetched entries
- Loop:
  - Fetch current STH until it changes and verify signature
  - Fetch all new entries. If they remain unavailable, count this as cheating.
  - Verify what you got

Auditoring

Auditors take partial information about a log as input and verify that this information is consistent with other partial information they have.

- might be an integral component of a TLS client
- might be a standalone service
- might be a secondary function of a monitor.


What about the Bad Guys?

“Misissued“ Certificates

- Can't be submitted to logs, because legitimate certificates already exist
- ***Thus, the maximum period of time in which the misissued certificate can be used is the allowed delay for a log to merge a cert into its tree.***

Bad Logs

Logs have two options of misbehaviour:

- Failing to incorporate a cert in the promised amount of time
- Presenting conflicting views of the tree

Lucky for us: Both cases are easily detectable, as long as someone runs a montior/auditor and clients do gossipping.


Onward, to the real world!

Current Logs and Clients

- Google's Pilot, Aviator and Rocketeer
- Digicert, Izenpe, Certly, Symantec, Wenafi, WoSign

- Chrome has CT built-in and enabled, Firefox is working on it

Monitors

- crt.sh by Comodo (feat. a query interface)
- RESTful monitor by tobermorytech
- Google's own implementations

Sources and Resources

certificate-transparency.org

RFC 6962

Mailing list: certificate-transparency@googlegroups.com

Code: github.com/certificate-transparency

Lots of pictures from different parts of the internet