COLLEGE OF SOCIAL & BEHAVIORAL SCIENCES
School of Information

# Integrity, Authenticity, and Privacy: Zero Knowledge Proof (ZKP)

Information Privacy with Applications

David Sidi (dsidi@email.arizona.edu)

# Warm-up

- Which application of PKC is seen in Schnorr’s ID protocol? (Hint: What is the use of the private key in the protocol?)

# Small mention of interesting things

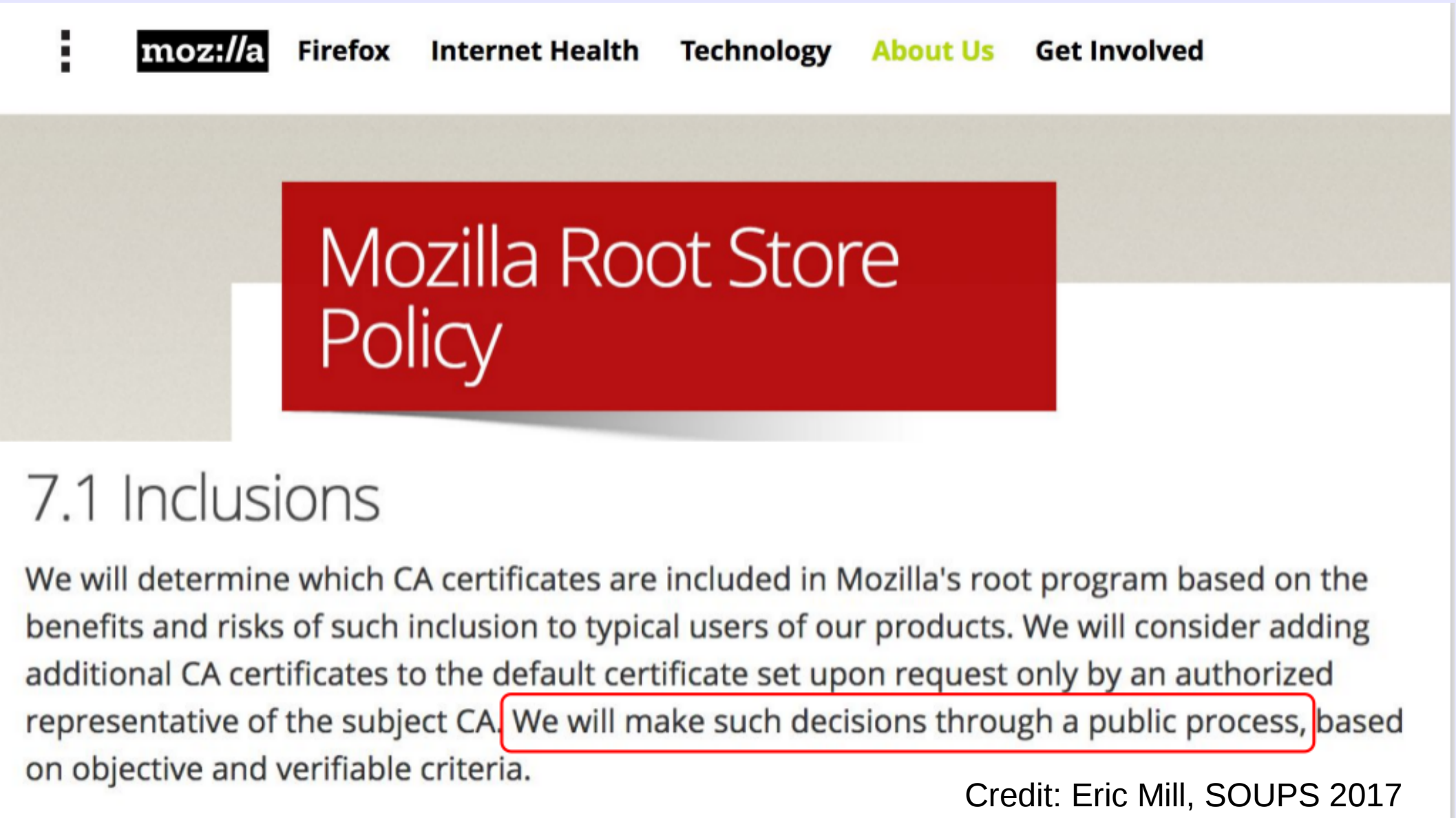
- http://qed-it.com/2017/07/challenge-one-the-functionality-of-zk-snark/
- Uber news
- I have not received any “proof of gpg key postings”
- Assignment 2

# Last time: Certificate Transparency

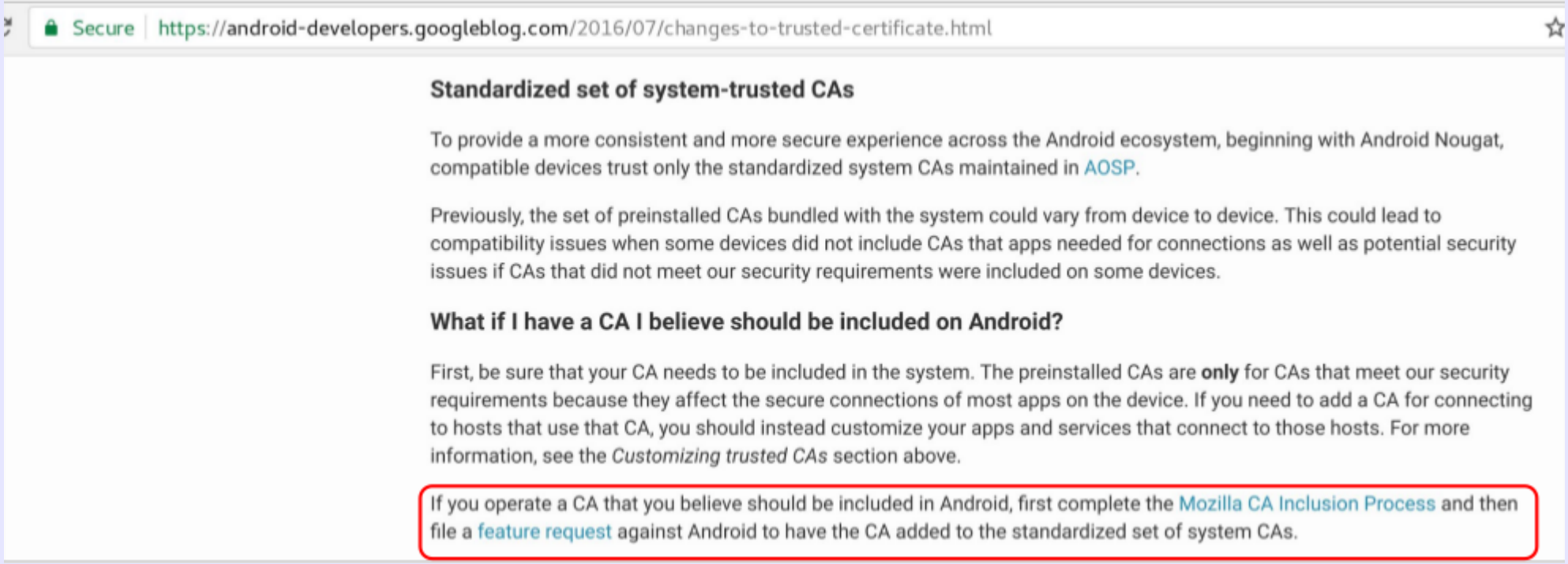
# SSL/TLS as soft authentication technology

- On the web in particular, we need a way to provide authenticated secure connections to entities (these also change often)
- Want it to require nothing of the user
- These two lead to a centralized system of Certificate Authorities

# There is a process for becoming a CA



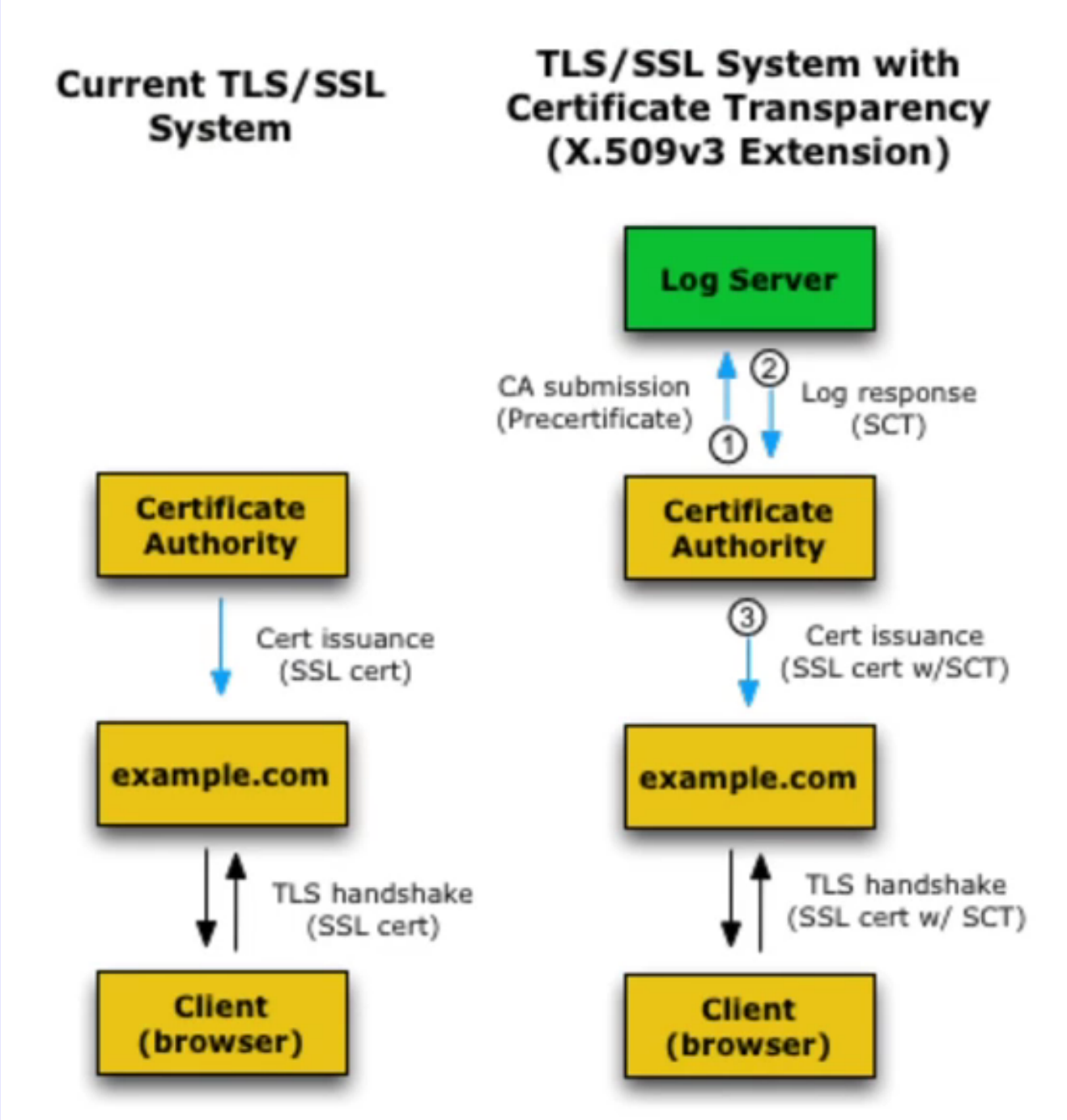

Credit: Eric Mill, SOUPS 2017




Credit: Eric Mill, SOUPS 2017

# Certificate Transparency

- Makes issuance of TLS/SSL certificates publicly auditable
  - cryptographically assured
  - append-only (no deletion, modification, or retroactive insertions)
  - public: log servers advertise their URL and public key
- Notice: **not** about whether the certificate is valid/revoked!
- Open source, anyone can run a log server
- Now *mandatory* for Firefox, Chrome, Opera (certificates only validate if they are logged)


Current TLS/SSL System
TLS/SSL System with Certificate Transparency (X.509v3 Extension)
Log Server
CA submission (Precertificate)
1
2
Log response (SCT)
Certificate Authority
Certificate Authority
3
Cert issuance (SSL cert)
Cert issuance (SSL cert w/SCT)
example.com
example.com
TLS handshake (SSL cert)
TLS handshake (SSL cert w/ SCT)
Client (browser)
Client (browser)

# Signed Certificate Timestamp

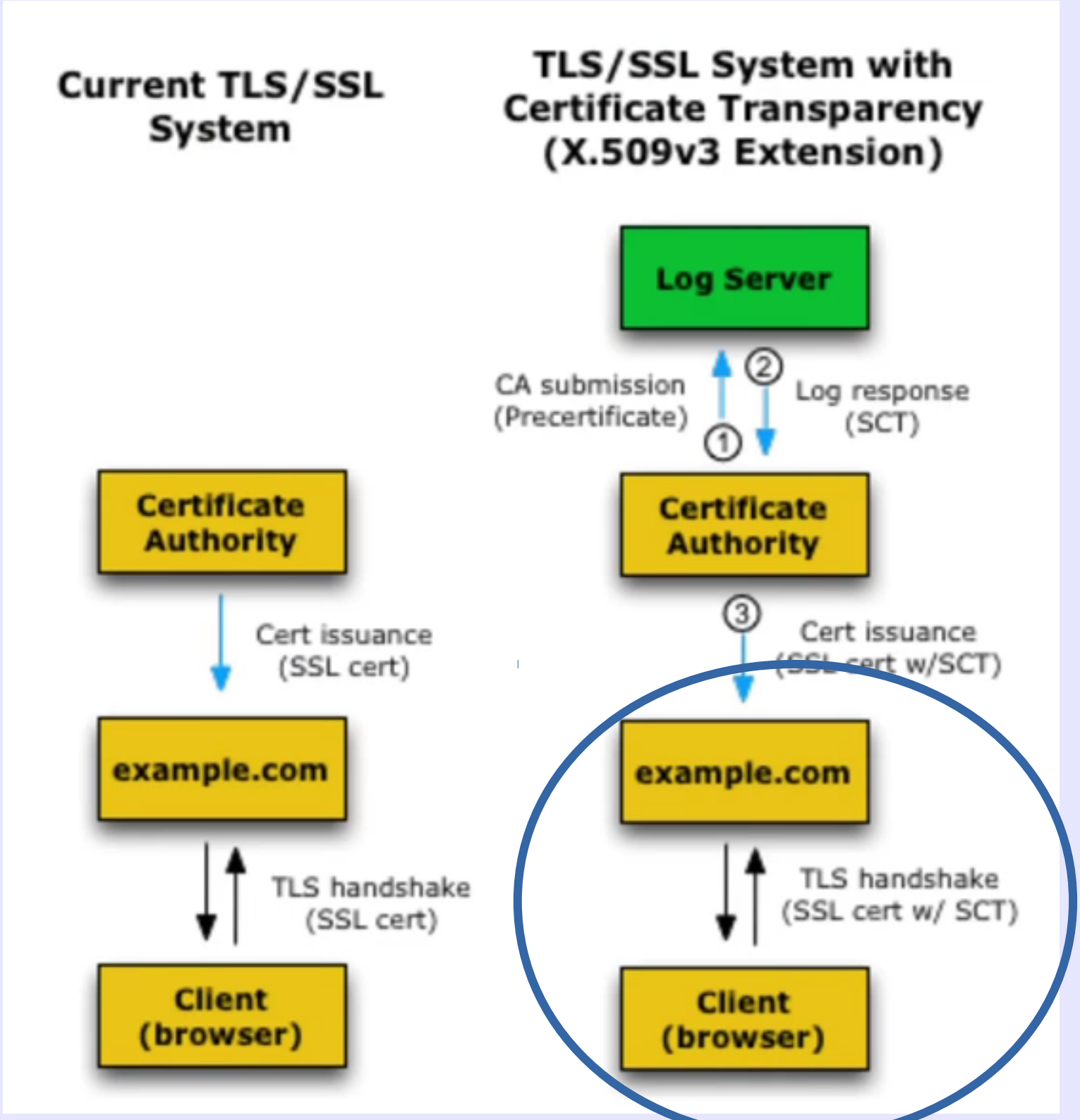
### 3.2. Structure of the Signed Certificate Timestamp

```
enum { certificate_timestamp(0), tree_hash(1), (255) }
  SignatureType;

enum { v1(0), (255) }
  Version;

  struct {
      opaque key_id[32];
  } LogID;

  opaque TBSCertificate<1..2^24-1>;

  struct {
    opaque issuer_key_hash[32];
    TBSCertificate tbs_certificate;
  } PreCert;

  opaque CtExtensions<0..2^16-1>;
```

- subject of the certificate's name
- issuer's name

- public key of the subject
- validity period
- version number and a serial number

RFC 5280 https://tools.ietf.org/html/rfc5280#section-4.1.2


Current TLS/SSL System
TLS/SSL System with Certificate Transparency (X.509v3 Extension)
Log Server
CA submission (Precertificate)
2
Log response (SCT)
1
Certificate Authority
Certificate Authority
3
Cert issuance (SSL cert)
Cert issuance (SSL cert w/SCT)
example.com
example.com
TLS handshake (SSL cert)
TLS handshake (SSL cert w/ SCT)
Client (browser)
Client (browser)

# Verification of an SCT is part of the TLS handshake

- An extension to the Online Certificate Status Protocol (OCSP) Stapling TLS protocol

```
$openssl s_client -connect sidiprojects.us:443 \
-tls1 -tlsextdebug -status
```

```
======================================
OCSP Response Data:
    OCSP Response Status: successful (0x0)
    Response Type: Basic OCSP Response
    Version: 1 (0x0)
    Responder Id: C = US, O = Let's Encrypt, CN = Let's Encrypt Authority X3
    Produced At: Nov 18 19:20:00 2017 GMT
    Responses:
    Certificate ID:
      Hash Algorithm: sha1
      Issuer Name Hash: 7EE66AE7729AB3FCF8A220646C16A12D6071085D
      Issuer Key Hash: A84A6A63047DDDBAE6D139B7A64565EFF3A8ECA1
      Serial Number: 0302CD2CAD56657A5F8E57DA8E5F0C1430A1
    Cert Status: good
    This Update: Nov 18 19:00:00 2017 GMT
    Next Update: Nov 25 19:00:00 2017 GMT
```

# OCSP stapling is better than the alternatives

- There are other ways for the client to verify the SCT

**3.2. Structure of the Signed Certificate Timestamp**

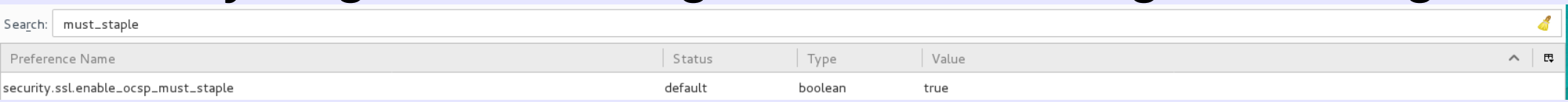
```
enum { certificate_timestamp(0), tree_hash(1), (255) }
  SignatureType;

enum { v1(0), (255) }
  Version;

  struct {
      opaque key_id[32];
  } LogID;

  opaque TBSCertificate<1..2^24-1>;

  struct {
    opaque issuer_key_hash[32];
    TBSCertificate tbs_certificate;
  } PreCert;

  opaque CtExtensions<0..2^16-1>;
```

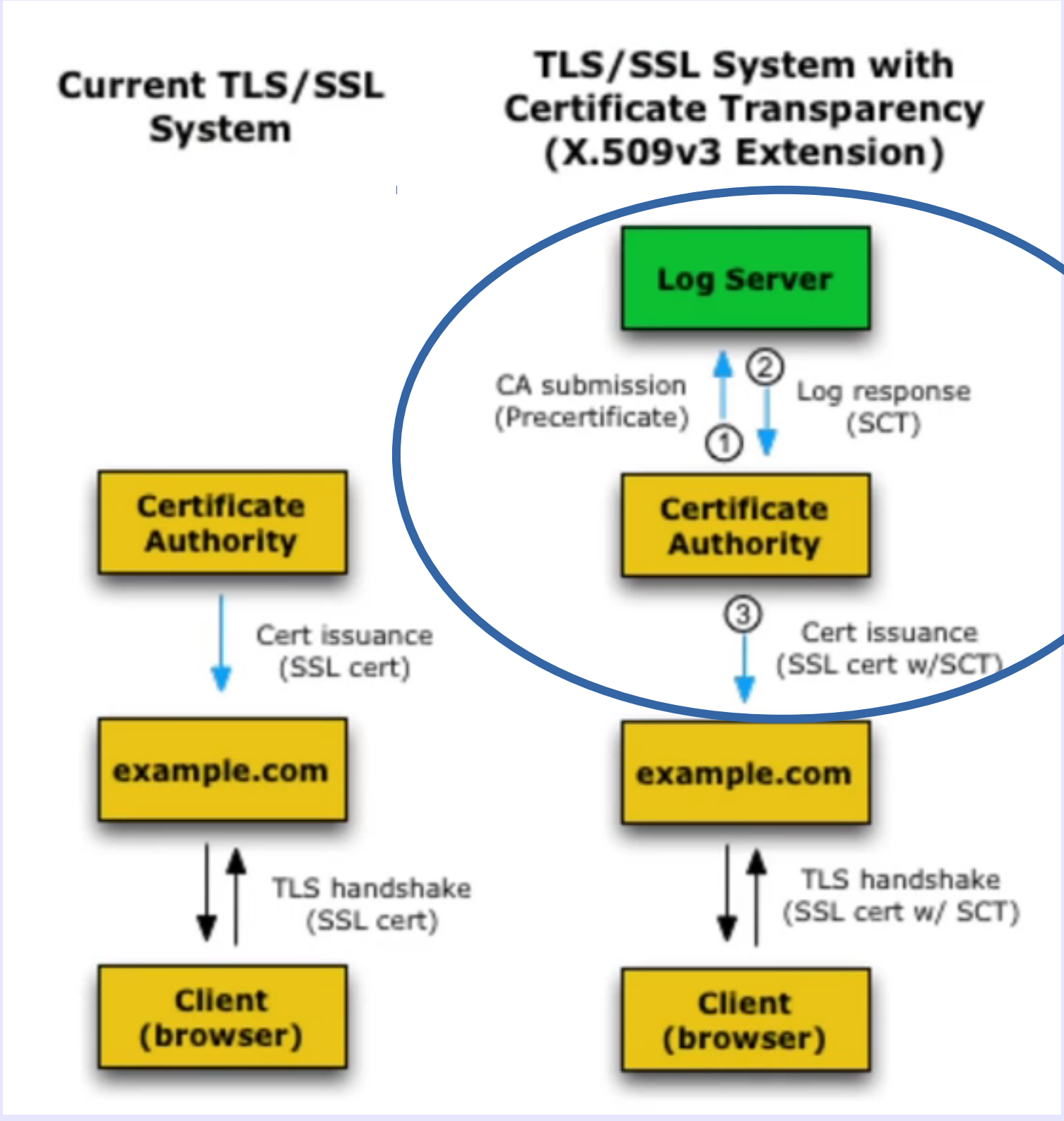
- subject of the certificate's name
- issuer's name

- public key of the subject
- validity period
- version number and a serial number
- **SignedCertificateTimestampList (as extension)**

# OCSP stapling is better than the alternatives

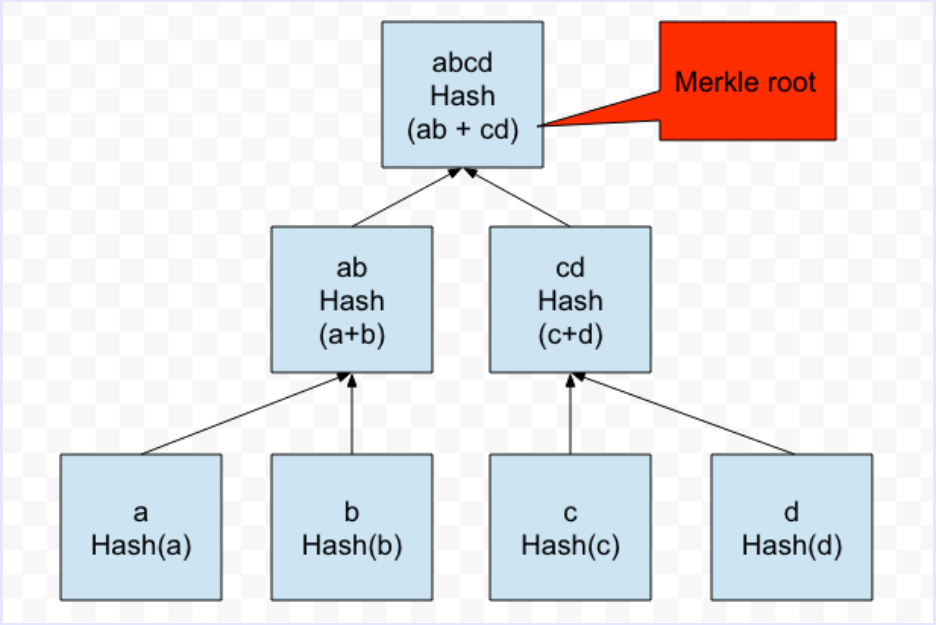
- There are other ways for the client to verify the SCT in the TLS handshake
- OCSP stapling does not require going out to the CA
  - the OCSP request, signed by the CA, is combined with the certificate and sent to the client
  - SCT can be included as part of this stapling
- Why might contacting the CA be a negative thing?




Current TLS/SSL System
TLS/SSL System with Certificate Transparency (X.509v3 Extension)
Log Server
CA submission (Precertificate)
2
Log response (SCT)
1
Certificate Authority
Certificate Authority
3
Cert issuance (SSL cert)
Cert issuance (SSL cert w/SCT)
example.com
example.com
TLS handshake (SSL cert)
TLS handshake (SSL cert w/ SCT)
Client (browser)
Client (browser)

# Log servers use Merkle Hash Trees to keep track of the certificates

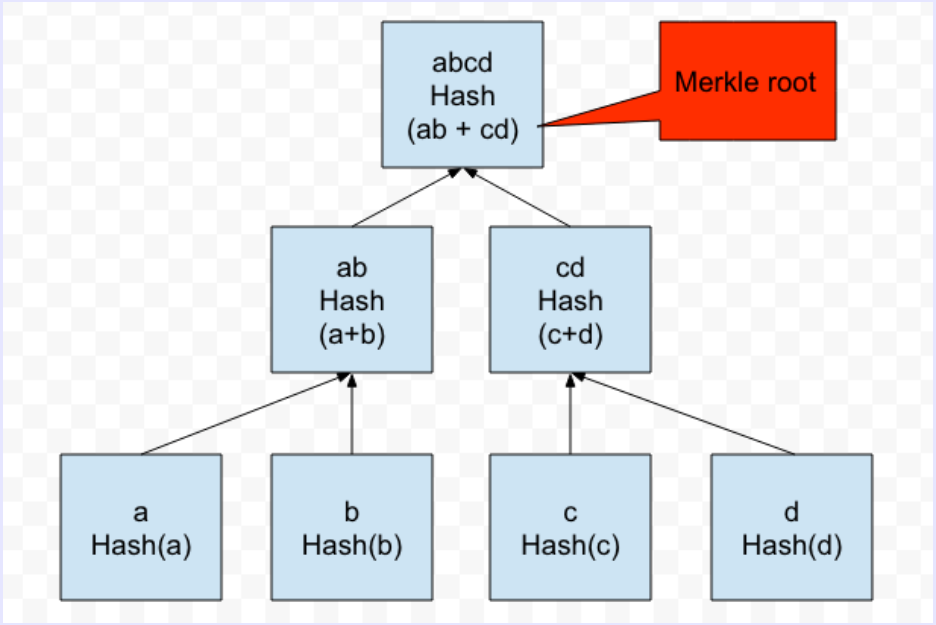
- Binary tree
- Calculated from the leaves: combine children's hashes to get the parent hash
- Can check integrity of a whole lot of hashes by checking one hash!
- All changes are auditable




credit

# Log servers use Merkle Hash Trees to keep track of the certificates

- Can catch CA's that are adding and removing illicit certificates
- Can catch cheating log servers
- Not enough to just calculate the root value to audit the log once new hashes are added. Why not?




credit

# Walk through: auditing a log addition

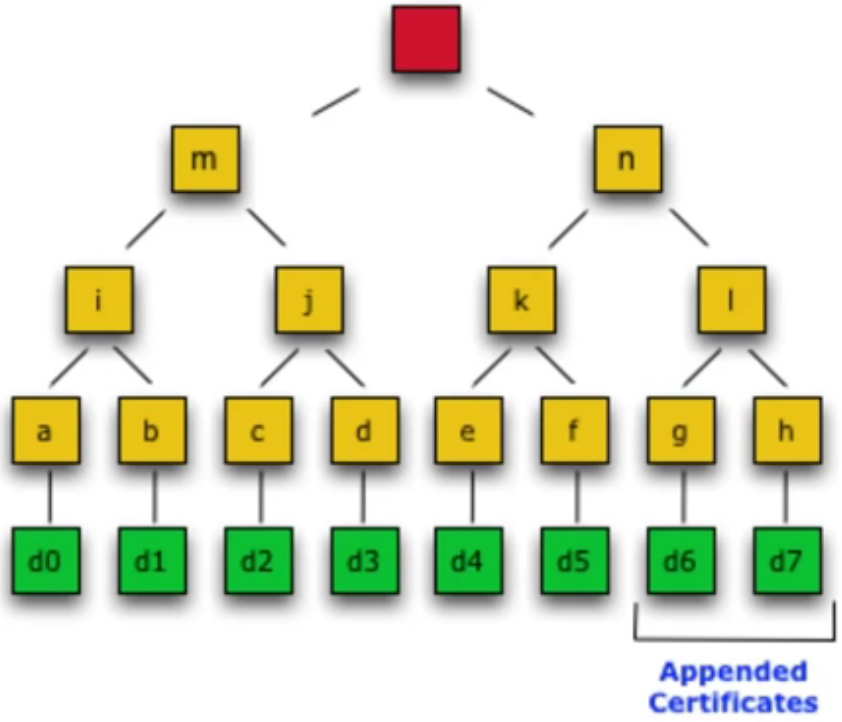



Figure 3

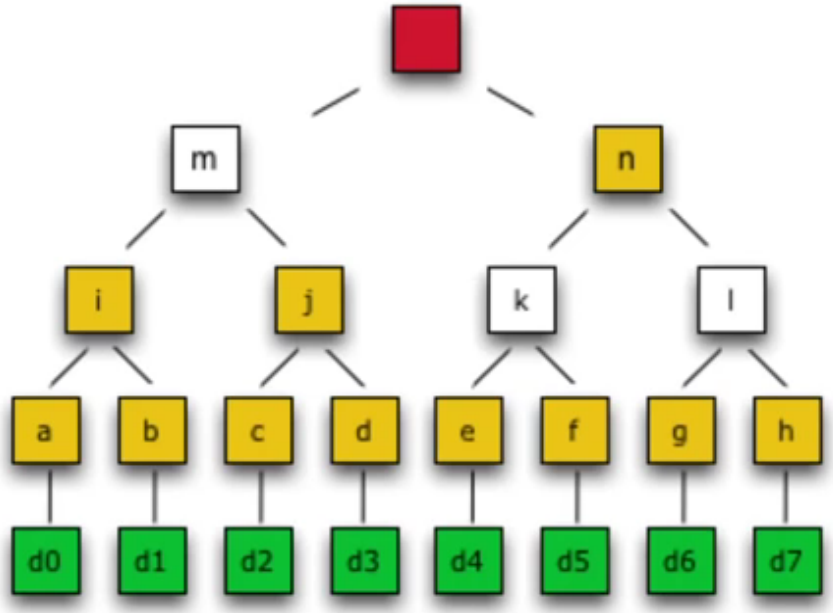
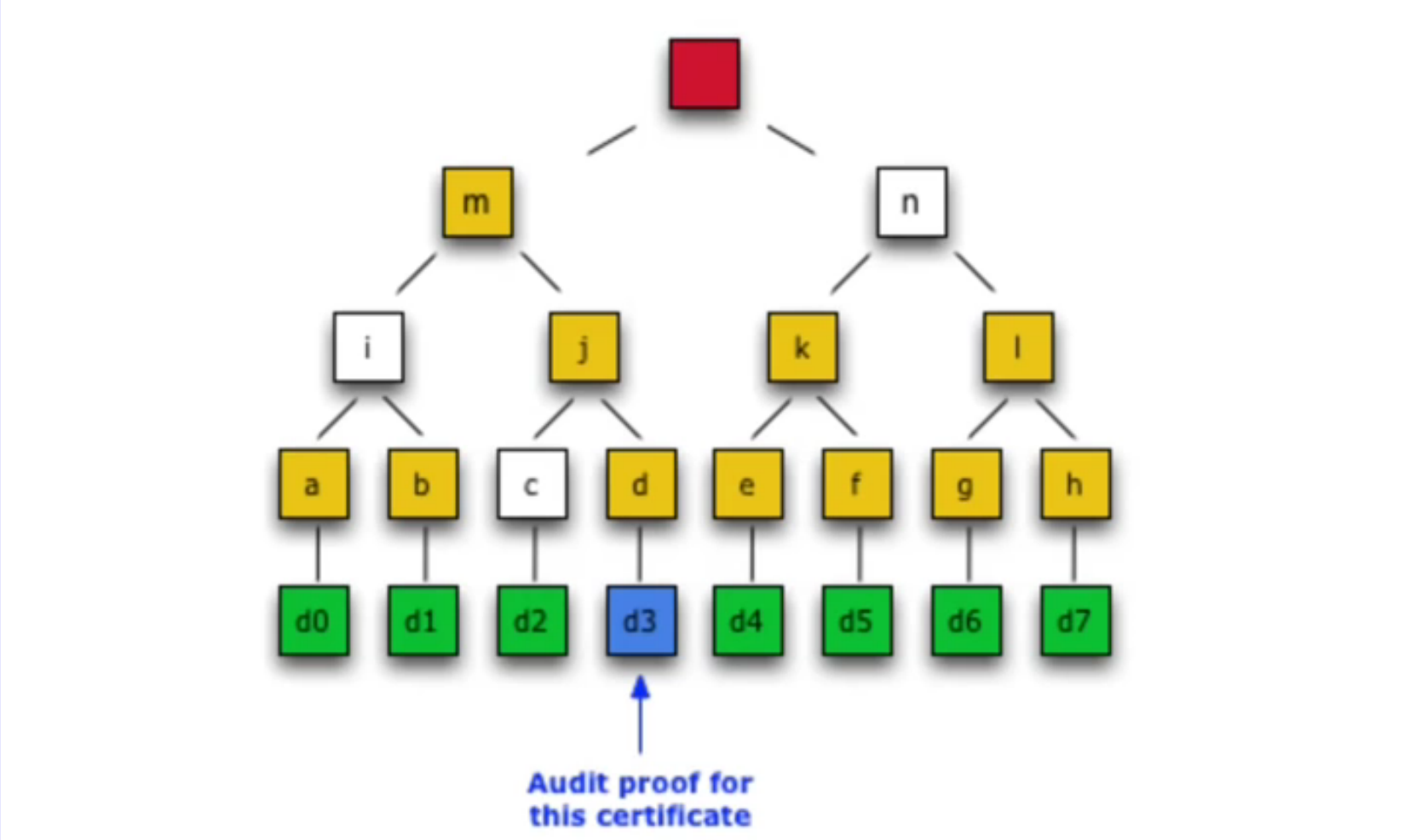



Figure 4

# Walk through: Auditing for presence of a particular certificate



# Log servers are still centralized in practice

- In theory, anyone can run a log
- In practice, there are only a few
  - Digicert: the first
  - Google: their idea; they run the big ones

```
$curl ct.googleapis.com/icarus/ct/v1/get-sth
{"tree_size":148531007,
"timestamp":1511196824947,
"sha256_root_hash":"bRmJZDeJZIs/WTOYZ3pA+MyJuOEZ9m+XGZIRU9fnViI=
",
"tree_head_signature":"BAMASDBGAiEAk+md3GDvKIPyuQ27UnLdDhKoVB5hn
zVDA8ZX1Dkx/JgCIQCDmYMAi6oqpAXk+LV/vIKwfrhyaCNrXl7N37moFv/BfA=="
}
```

- Use crt.sh to search manually from the browser. Certspotter can help you monitor your domains (https://sslmate.com/certspotter/)

# About those temporary certificates that Schmiedecker mentions

## Sustaining Digital Certificate Security

October 28, 2015

Posted by Ryan Sleevi, Software Engineer

*This post updates our previous notification of a misissued certificate for google.com*

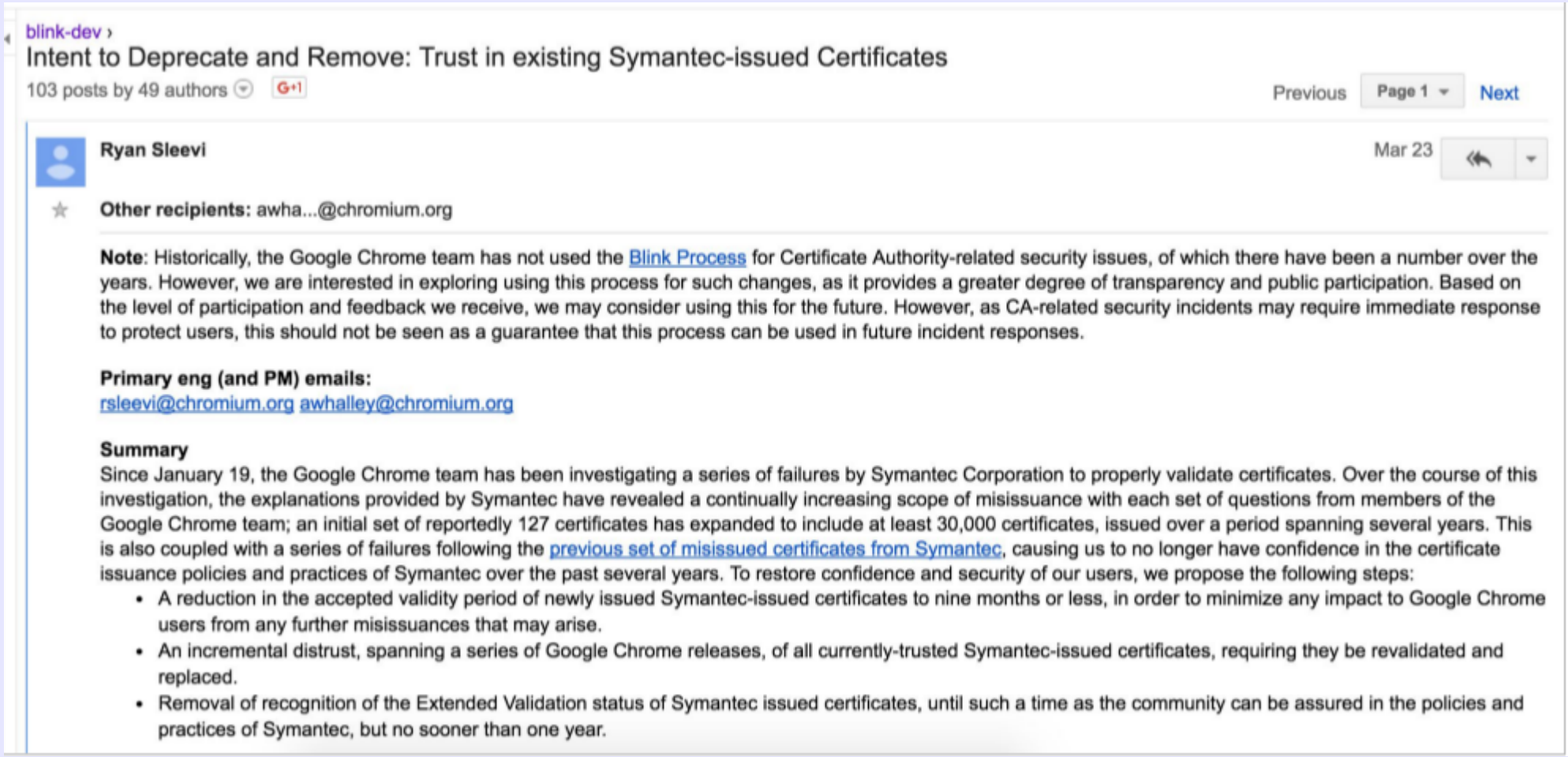
Following our notification, Symantec published a report in response to our inquiries and disclosed that 23 test certificates had been issued without the domain owner's knowledge covering five organizations, including Google and Opera.

However, we were still able to find several more questionable certificates using only the Certificate Transparency logs and a few minutes of work. We shared these results with other root store operators on October 6th, to allow them to independently assess and verify our research.

Symantec performed another audit and, on October 12th, announced that they had found an additional 164 certificates over 76 domains and 2,458 certificates issued for domains that were never registered.



Credit: Eric Mill, SOUPS 2017



blink-dev ›

Intent to Deprecate and Remove: Trust in existing Symantec-issued Certificates

103 posts by 49 authors G+1

Previous Page 1 Next

**Ryan Sleevi** Mar 23

**Other recipients:** awha...@chromium.org

**Note**: Historically, the Google Chrome team has not used the Blink Process for Certificate Authority-related security issues, of which there have been a number over the years. However, we are interested in exploring using this process for such changes, as it provides a greater degree of transparency and public participation. Based on the level of participation and feedback we receive, we may consider using this for the future. However, as CA-related security incidents may require immediate response to protect users, this should not be seen as a guarantee that this process can be used in future incident responses.

**Primary eng (and PM) emails:**
rsleevi@chromium.org awhalley@chromium.org

**Summary**
Since January 19, the Google Chrome team has been investigating a series of failures by Symantec Corporation to properly validate certificates. Over the course of this investigation, the explanations provided by Symantec have revealed a continually increasing scope of misissuance with each set of questions from members of the Google Chrome team; an initial set of reportedly 127 certificates has expanded to include at least 30,000 certificates, issued over a period spanning several years. This is also coupled with a series of failures following the previous set of misissued certificates from Symantec, causing us to no longer have confidence in the certificate issuance policies and practices of Symantec over the past several years. To restore confidence and security of our users, we propose the following steps:

- A reduction in the accepted validity period of newly issued Symantec-issued certificates to nine months or less, in order to minimize any impact to Google Chrome users from any further misissuances that may arise.
- An incremental distrust, spanning a series of Google Chrome releases, of all currently-trusted Symantec-issued certificates, requiring they be revalidated and replaced.
- Removal of recognition of the Extended Validation status of Symantec issued certificates, until such a time as the community can be assured in the policies and practices of Symantec, but no sooner than one year.

Credit: Eric Mill, SOUPS 2017

# Other ways to fix TLS

- Using GPG, with monkeysphere
  - http://web.monkeysphere.info/
- Flexible trust model of WoT used for PKI
- Problem: goes out to the keyserver for failing requests

# Extending CT: Trillian

- Generalizing the Merkle Tree datastructure that CT log servers use, creating a *transparent store*

## Overview

Trillian is an implementation of the concepts described in the Verifiable Data Structures white paper, which in turn is an extension and generalisation of the ideas which underpin Certificate Transparency.

Trillian implements a Merkle tree whose contents are served from a data storage layer, to allow scalability to extremely large trees. On top of this Merkle tree, Trillian provides two modes:

- An append-only **Log** mode, analogous to the original Certificate Transparency logs. In this mode, the Merkle tree is effectively filled up from the left, giving a *dense* Merkle tree.
- A **Map** mode that allows transparent storage of arbitrary key:value pairs. In this mode, the key's hash is used to designate a particular leaf of a deep Merkle tree, giving a *sparse* Merkle tree. (A Trillian Map is an *unordered* map; it does not allow enumeration of the Map's keys.)

Note that Trillian requires particular applications to provide their own personalities on top of the core transparent data store functionality; example code for a certificate transparency log and for a log-derived map are included to help with this.

https://github.com/google/trillian

# Zero-Knowledge Proof (ZKP)

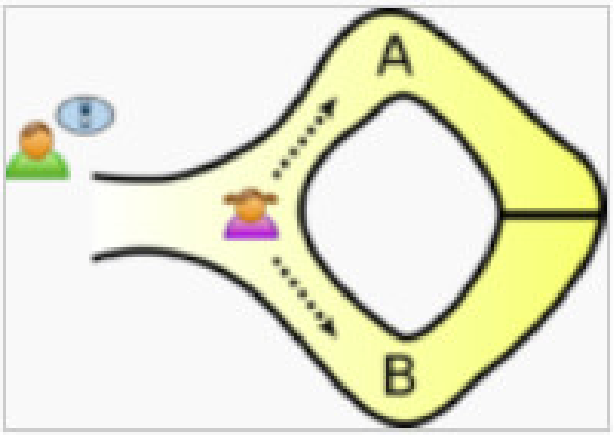
# Where’s Waldo

- Let’s build intuition about what ZKP does with an example from Naor, Naor, and Reingold
- Demo (with the Elmo)

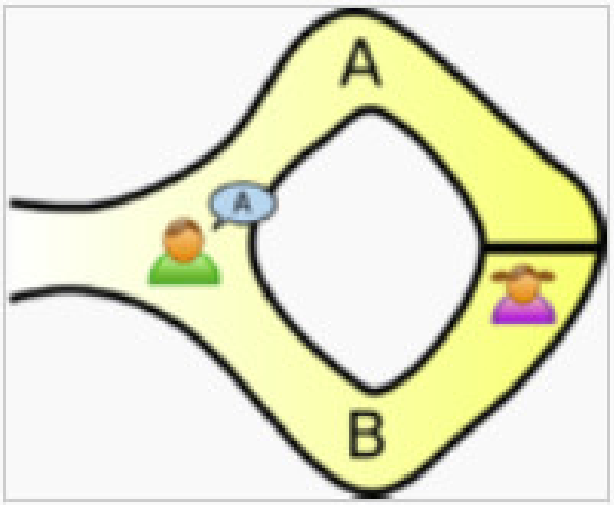
# Alibaba’s cave

- There must be a law somewhere that this example be discussed
- From Quisquater, Guillou and Berson

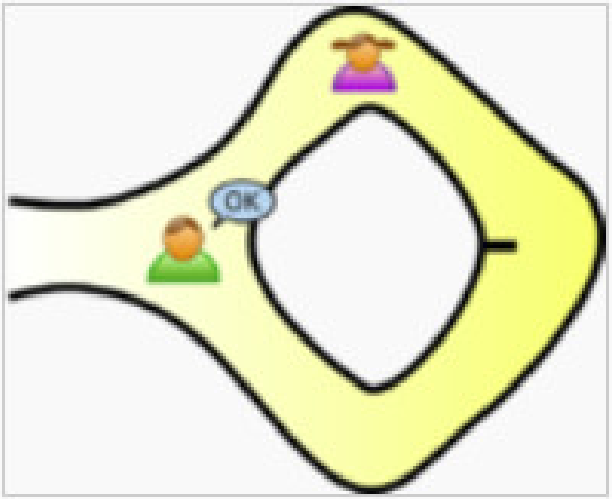

https://www.artstation.com/artwork/3b3g2




Peggy randomly takes either path A or B, while Victor waits outside




Victor chooses an exit path




Peggy reliably appears at the exit Victor names

Question: Should running this be convincing to Victor?

# ZKP establishes two things

- There are two facts proved with ZKP
  - existence of a solution to a problem (the solution set is nonempty)
  - knowledge of **a** solution (a member of the solution set)
- Problems are in NP

# ZKP has a coherence property

- False sentences can’t be proved by a cheating prover to an honest verifier, except with small probability (soundness*)
- True sentences can be proved by honest provers to honest verifiers (completeness)
- notice we can move between sentences and problems

# ZKP has a zero-knowledge property

- For a true sentence **s**, a cheating verifier learns nothing beyond the truth of **s** (privacy / zero-knowledge)

# Schnorr’s Identification Protocol

- Schnorr’s ID protocol is a pair challenge-response protocol
- Relies on the discrete log assumption
- Others rely on RSA assumption (Fiat-Shamir)

# ZKP identification protocols let Peggy prove that she knows a secret to Victor

- In symmetric cryptosystems, the secret is known by Victor too; in public key cryptosystems it is not
- But wait! For the Schnorr case, isn’t there an easy way to do that? Could we just encrypt to the intended recipient, and have them respond to us with what we said, encrypted to us?
- There is a better way

# Key authentication centers have a role in ZKP

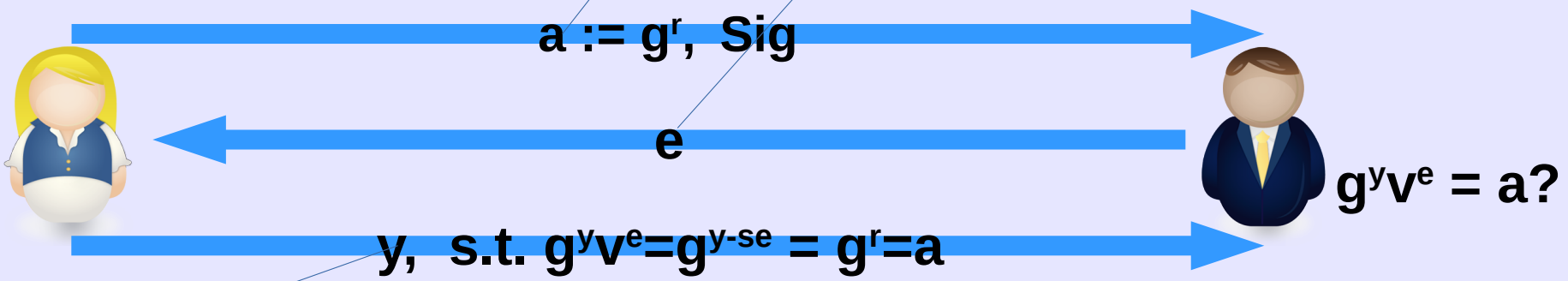
- A key authentication center (KAC) can sign a public key to certify that the key belongs to the person who claims it
  - Schnorr does this with an identification string I and a public key v
  - KAC signs the pair (I,v)
- We have a name for these signed public keys: what is it?

# Schnorr Identification Protocol

Publicly choose:
- a large prime **p**, the group order
- a generator **g** for the group

Secretly generate:
- Peggy randomly chooses an integer secret **s**
- Peggy randomly chooses an integer witness **r**
- Victor randomly generates an integer challenge **e**

Let Sig be a signature of (**I**, **v**) for identification string I and public key **v** := $g^{-s}$



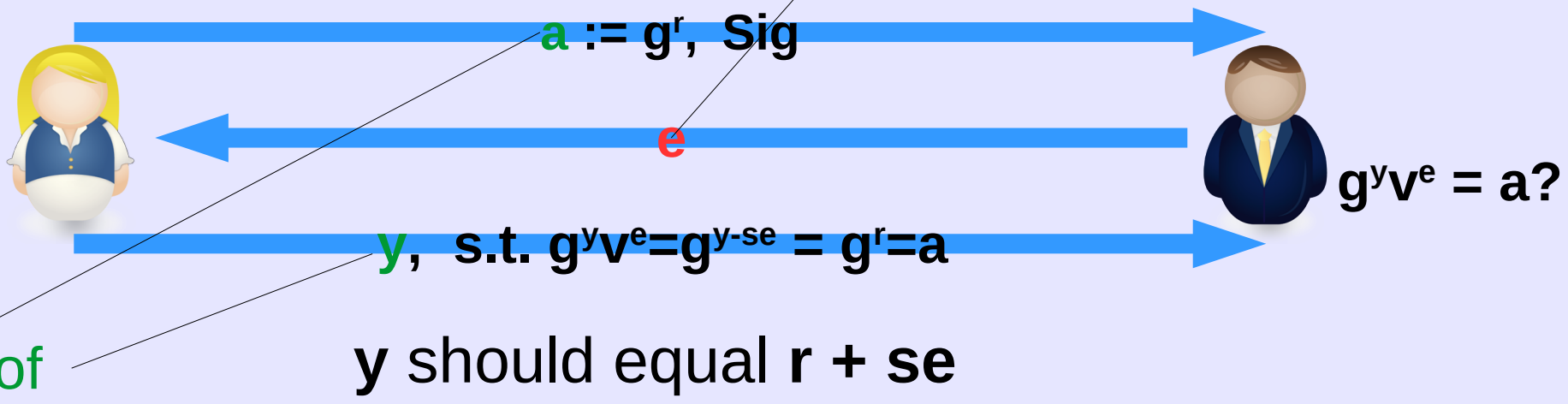

**y** should equal **r + se**

# Schnorr Identification Protocol

Publicly choose:
- a large prime **p**, the group order
- a generator **g** for the group

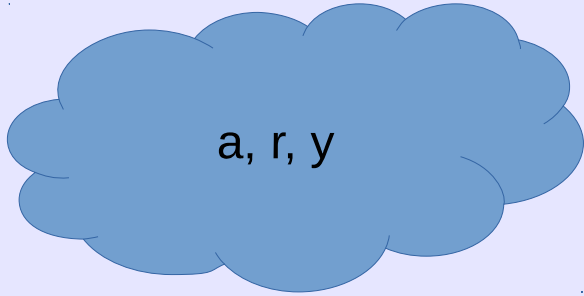
Secretly generate:
- Peggy randomly chooses an integer secret **s**
- Peggy randomly chooses an integer witness **r**
- Victor randomly generates an integer challenge **e**

Let Sig be a signature of (**I**, **v**) for identification string I and public key **v** := $g^{-s}$



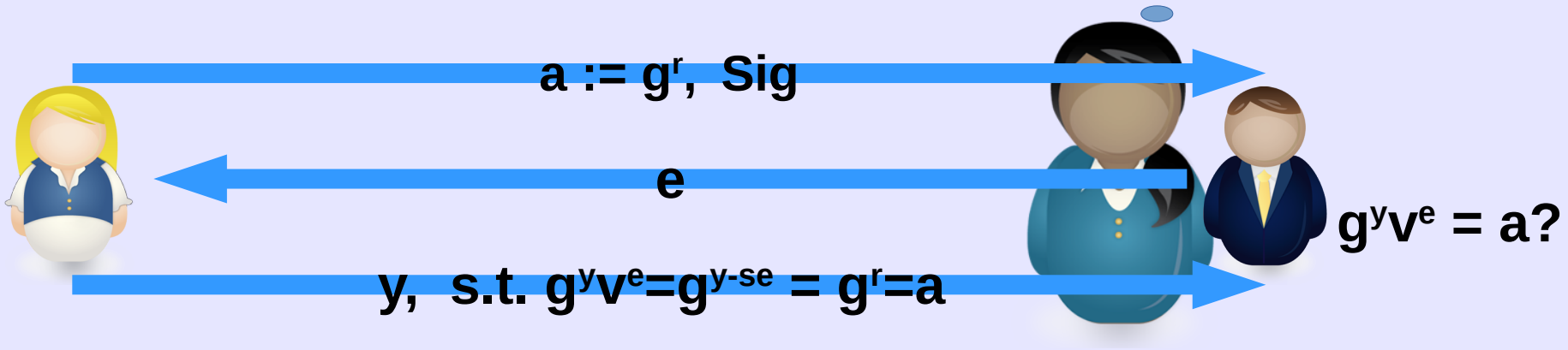
# Schnorr Identification Protocol: Honest verifier

Publicly choose:
- a large prime **p**, the group order
- a generator **g** for the group




Secretly generate:
- Peggy randomly chooses an integer secret **s**
- Peggy randomly chooses an integer witness **r**
- Victor randomly chooses an integer challenge **e**

Let Sig be a signature of (**I**, **v**) for identification string I and public key $\mathbf{v} := \mathbf{g}^{-s}$



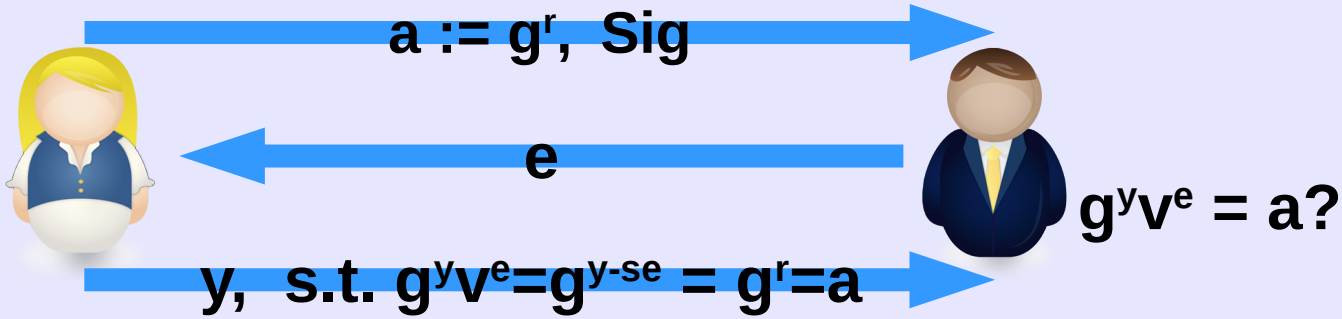

**y** should equal **r + se**

# Schnorr Identification Protocol: Honest verifier

Publicly choose:
- a large prime **p**, the group order
- a generator **g** for the group

Secretly generate:
- Peggy randomly chooses an integer secret **s**
- Peggy randomly chooses an integer witness **r**
- Victor randomly chooses an integer challenge **e**

Let Sig be a signature of (**I**, **v**) for identification string I and public key **v** := $g^{-s}$




**y** should equal **r + se**

$(e,y,a)$ and $(\bar{e},\bar{y},\bar{a})$ are identically distributed

# Schnorr Identification Protocol: Crooked prover (PK-Only)

Publicly choose:

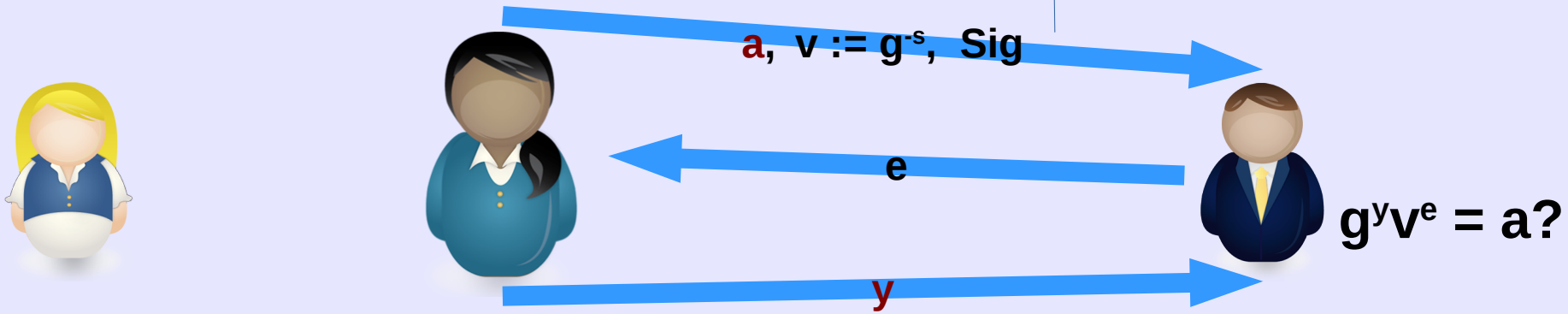
- a large prime *p*, the group order
- a generator g for the group

Secretly generate:

- Peggy randomly chooses an integer secret s
- Peggy randomly chooses an integer witness r
- Victor randomly chooses an integer challenge c

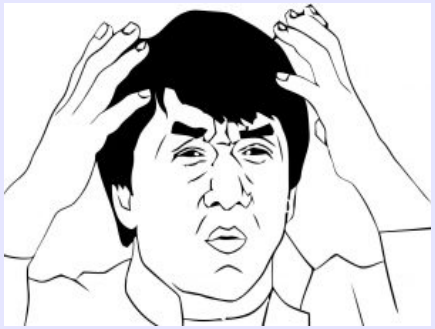
Let Sig be a signature of (I, v) for identification string I and public key v

Still Peggy's public key




Probability of success is negligible for Mallory, if DL problem is hard. Proof is by contraposition

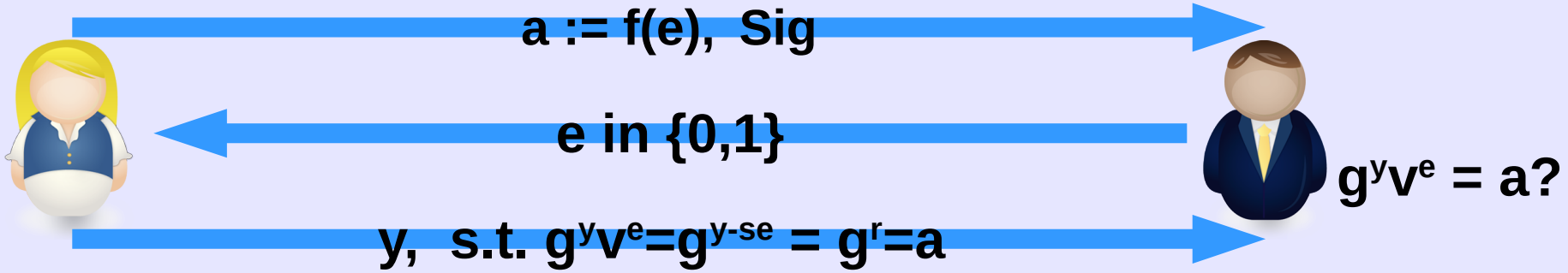
# Plot twist!



- Schnorr’s identification protocol as given is not a ZKP

- We looked at honest verifiers and cheating provers, but the problem comes with malicious verifiers
- Okamoto Identification Scheme is secure against active attacks

# Fixing it up by restricting the challenge set




- ZKP requires limiting the size of the challenge set. What about soundness?