TOPIC: 6.857 Lecture Notes

DATE: 4/30/18

FILE UNDER:

PAGE: L21.1

**Admin:**

Project presentations start Monday May 7th

Final report is due Wednesday May 16th

Boaz Barak on ZKP Wed this week

**Today:**

Public-Key Infrastructures (PKI)

- ☐ Keys, names, parties
- ☐ Certs (X.509)
- ☐ Identity-Based Encryption (IBE)
- ☐ BIP-32
- ☐ SPKI/SDSI

Open Problems

**Reading:**

Ferguson, Part IV

Smart, Chapter 18

TOPIC:

DATE:

FILE UNDER:

PAGE:

## Keys, Names, Parties

$(PK, SK) \leftarrow KeyGen(1^\lambda)$ for PK scheme

Distribution & Authentication of PKs

Alice: What is Bob's PK?

name

Diffie-Hellman: Directory mapping

names $\rightarrow$ (public) keys

How to implement such a directory?

Do we need a TTP? (Trusted 3rd party)

How to revoke a key? (e.g. after compromise of SK)

Types of PKs: signing

encryption

certificate authority

TOPIC:

DATE:

FILE UNDER:

PAGE:

# Names

Identifier, string.

Fixed-length, variable length. Alphabet?

Human-friendly.

Memorable

May have semantics "IBM", "John Smith". La
na

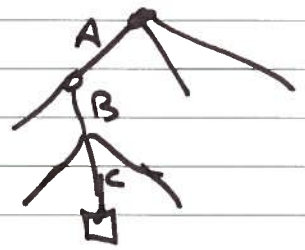
Name of key? of person? of org?

Globally unique? Locally unique? Not nec. unique?

Relative to some context or namespace

May be segmented/hierarchical

A.B.C




Nick names.

Pseudonyms.

PK as name? (cf. Bitcoin)

(tries to avoid problem of name mgt.)

TOPIC:

DATE:

FILE UNDER:

PAGE:

## Parties

People

Computers, Servers,

Organizations

Users.

Trusted Third Parties (TTPs).

TOPIC:

DATE:

FILE UNDER:

PAGE:

# Certificates

Signed stmt from an "issuer"

about a "subject" (name)'s public key

(Loren Kohnfelder, MIT B.S. thesis, 1978)

- one entry extracted from master directory, signed.
- "Alice Jones' PK is (RSA, n=314..., e=57)."
  (signed, CA) (dated)
- CA = "certificate authority"

  Why is a CA trusted? By whom?

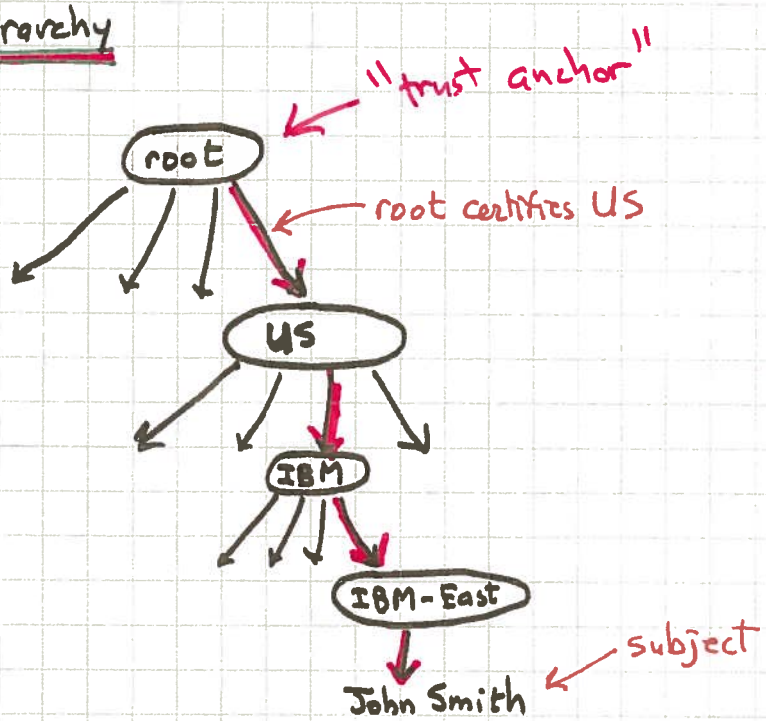
  For what names?
- There can't be one CA...

Alice can learn Bob's PK from his certificate.

(which Bob can give her)

(example: server certificates)

# X.509 hierarchy




DN = "distinguished name"

= "CO=US/ORG=IBM/DIV=IBM-EAST/CN=John-Smith"

subject's DN

These DN's become unwieldy for people to use.

## Certificate chain

TOPIC:

DATE:

FILE UNDER:



## Certificate fields:

- Version #
- Cert serial #
- Signature algorithm
- Issuer DN
- Subject DN
- Validity period (not before, not after)
- Subject PK (alg & key)
- Issuer unique #
- Extensions: type & critical/non-critical flag
  - ↳ key usage (encs, sigs, certs)
  - cert policies
  - subject alternate name
  - path constraints

Issuer PK missing?

Used by TLS (SSL)

TOPIC:

DATE:

FILE UNDER:



## Certificate revocation

Why? – key compromise
– change of affiliation
– change of authorization
– change of name (e.g. merger)

Fairly high "churn rate"

If certificate says "good until 2020-12-01"
who decides if that is good enough?
issuer? (current practice)
relying party? ⟸ Should be relying party!
(they are taking the risk...)

Helpful to think about it this way:

- Issuer maintains authoritative DB
- Certificates are merely "snapshots" of items
- Note that DB may not fully reflect key compromises, etc...

Method 1: On-line check:

Relying party asks issuer if cert still good

Issuer gives signed response (new cert?)

≡ OCSP (online certificate status protocol)

Puts heavy load on server!

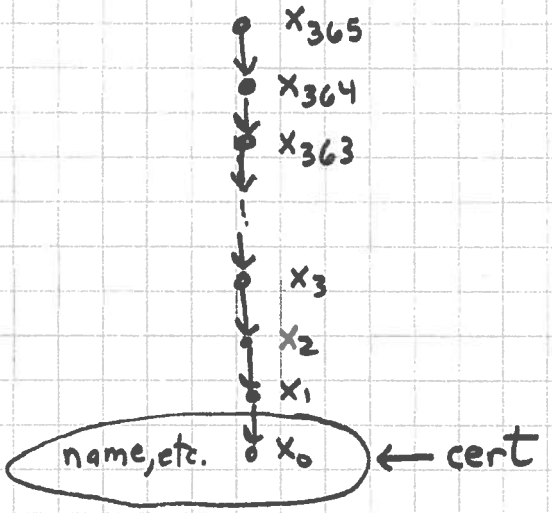
**Method ②:** CRL's (certificate revocation list)

Server periodically issues CRL, giving list of revoked cert serial #'s (signed, of course)

CRL can get long!

**Method ③:** (due to Micali)

cert contains end point of a hash chain




On day $d+i$, where $d$ = cert issuing date, you need $x_i$ to validate cert. Server can give $x_i$ to keyholder, or to anyone else. Relying party hashes $i$ times & checks that result $= x_0$

If no $x_i$ given out, cert "expires".

TOPIC:

DATE:

FILE UNDER:

PAGE:

# IBE - Identity-Based Encryption

User PK = (TTPkey, username)

TTP issues SK to user.

(TTP knows user's SK)

## Boneh-Franklin

- $K_{TTP} = s \cdot P$ $\quad$ s = TTP SK, P = generator of G
- User with ID wants SK

$$Q_{ID} = H_1(ID) \in G^*$$

$$d_{ID} = s \cdot Q_{ID}$$ user SK

- Enc m to user using $(K_{TTP}, ID)$

$$Q_{ID} = H_1(ID) \in G^*$$

$$g_{ID} = e(Q_{ID}, K_{TTP})$$

$$c = (r \cdot P, m \oplus H_2(g_{ID}^r)) = (u, v)$$

- Decryption given $(u, v)$

$$m = v \oplus H_2(e(d_{ID}, u))$$

Extend to HIBE

hierarchical

# SPKI/SDSI

- alternate framework by Ellison, Lampson, Rivest & others
- No global names (sort of...)
- Each PK is a CA & has its own name space

$\boxed{K.A}$ is really a global name

identifier or name

PK of issuer

- Two types of certificates:

  name cert = associates name with key or another name (!)

  auth cert = gives permission (authorization) to subject(s)

- Name certs:

  Provide elegant & flexible "algebra" of names

  Allows one to describe groups (multivalued names)

cert = (K, A, X)

$K.A \Rightarrow X$

give as "rewrite rule"

K = issuer PK
A = identifier (name)
X = PK (key)
or PK.id (name)
or $\underbrace{PK.id_1.id_2 \cdots .id_k}_{\text{extended name}}$

## Authorization certs & ACL's

ACL = access control list

Can put name on ACL for a resource:

"Only individuals in K. friends may read files in this directory"

**Authorization cert:**

Issuer key K

Rights being granted [read directory foo]

Delegatable or not

Subject key or name

K [read "/foo"; delegatable] ⇒ K. friends

There are polynomial-time algorithms for determining whether a given collection of name & auth certs implies that a given key is authorized to perform a given action

**Examples:** (let K be my PK)

$K.Bob \Longrightarrow K_0$ "I say Bob's PK is $K_0$"

(Note: the name "Bob" is my choice & arbitrary. It doesn't need to correlate with anything else...)

$K.Bob \Rightarrow K_1.Bob\text{-}Smith$ "My Bob is the same as $K_1$'s Bob Smith"

$K.Bob \Rightarrow K.mit.Bob\text{-}Smith$ "My Bob is same as mit's Bob Smith

$K.mit \Rightarrow K_2$

Inference

$K.Bob \Rightarrow K_2.BobSmith$ inferred rule

Groups: $K.friends \Rightarrow K.Alice$
$K.friends \Rightarrow K.Bob$
} group members

...

Let $K_2$ = mit PK

$K_2.faculty \Rightarrow K_2.eecs.faculty$

$K_2.faculty \Rightarrow K_2.math.faculty$

⋮

(group defined as union of other groups)

TOPIC:

DATE:

FILE UNDER:

PAGE:

# BIP-32 (Bitcoin Improvement Proposal)

User-generated tree of PKs. Unlinkable. Can generate child PK from parent PK without knowing SK.

Actually, (PK, chaincode) pair
random #

(PK', cc')

i = child #

$$\underbrace{(PK, cc)}_{\text{parent}} \xrightarrow{i} \underbrace{(PK', cc')}_{\text{child (not hardened)}}$$

$$\underbrace{(SK, cc)}_{\text{parent}} \xrightarrow{i} \underbrace{(SK', cc')}_{\text{child}}$$

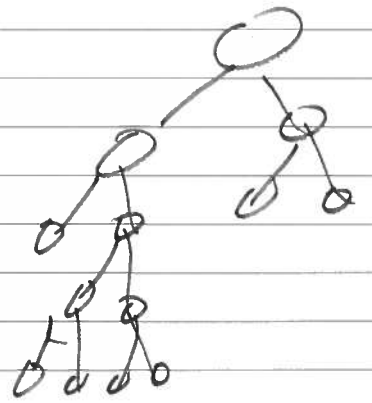
i = child #

.-----> not possible

Two kinds of children: hardened, & not

derive (PK', cc') from cc, PK, i

or cc, SK, i (hardened)



← leaves are hardened
can receive payments
can't make them.

TOPIC:

DATE:

FILE UNDER:

PAGE:

# Open Problems

1. Post-Quantum Crypto
2. Resilience
3. Assurance
4. Crypto Policy