6.857 Rivest
4/4/11 L16.1

Admin: Quiz on Wed (4/6/11)

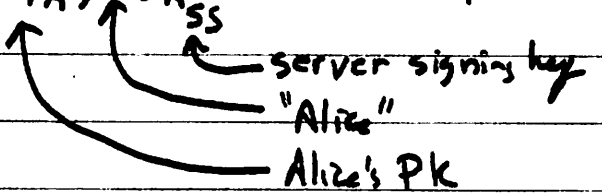
Open notes only. (No books, laptops, smartphones, ...)

Outline:

- ☐ certificates
- ☐ scaling
- ☐ X.509
- ☐ SPKI/SDSI
- ☐ revocation
- ☐ IBE

Certificates:

- Last time we did Needham-Schroeder
- $\{K_{PA}, A\}_{K_{SS}}$ is a prototypical certificate:



S certifies that Alice's key is K_{PA}

- Others can get this from S, or from A.

Scaling

- How do we go from 100 users to 10^8 users?
- Everything starts breaking:
  - there is no one server everyone trusts (?)
  - one server can't handle load
  - what are <u>names</u>?? ⇐ subtle, but hard & important

- Names:

  How does Alice know Bob's name?

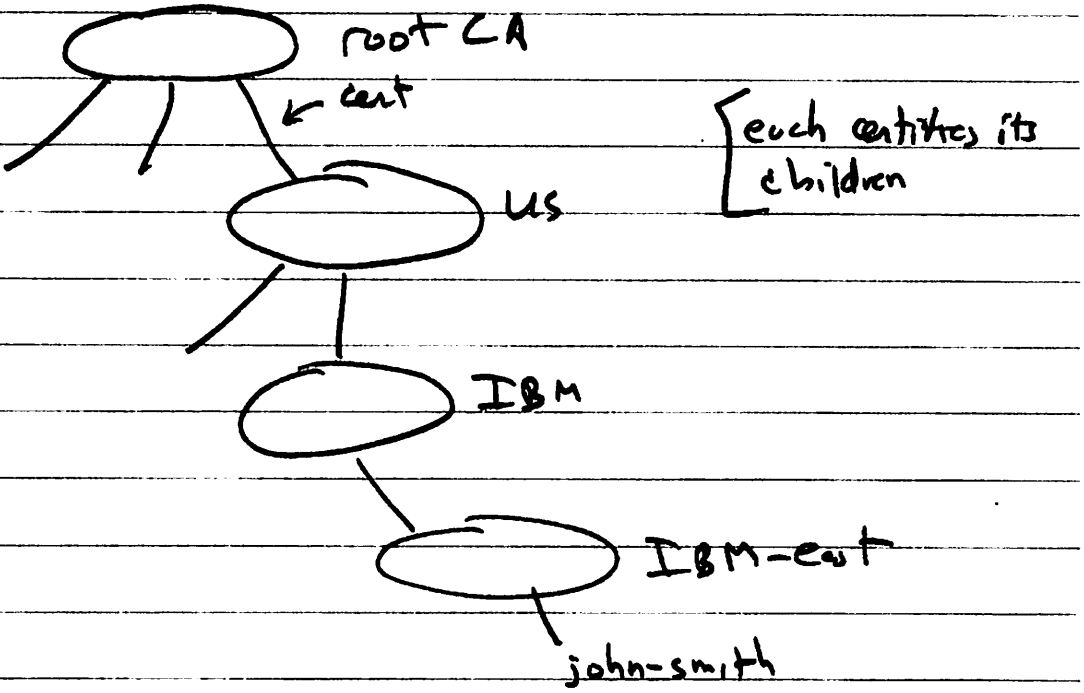
  Who guarantees that names are unique? How is this done?

  (compare: email addresses...)

  If Alice can get Bob's (email name or address) correctly, why can't she get his PK the same way?

6.857 Rivest
4/1/09 L17.3

X.509 hierarchy



DN = "distinguished name" =

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

names become unwieldy for people to use

Certs have: version #
cert serial #
sig. alg.
Issuer DN
Subject DN
validity period
subject PK alg & key
Issuer unique #
" " "
extensions: type ; crit/non-crit/value
key usage (enc/sig, cert sig)..
subject alt name. path constraints

6.857 Rivest
4/8/09 L17.4

SPKI/SDSI

- no global names
- each PK has its own name space (each key is a CA...)
- certs have validity period
- two types of certs: name & auth

name cert: (K is issuers PK)

$K_{\bullet}$ Alice $\Rightarrow$ value (signed by K)

value $\rightarrow$ K' (another PK)

or $\rightarrow$ another name

e.g. $K_{\bullet}$ Bob $\Rightarrow$ [redacted] K_0

$K_{\bullet}$ Bob $\Rightarrow$ K_1 • Bob-Smith

$K_{\bullet}$ Alice $\Rightarrow$ K_2 • eecs • Alice-Smith

K_2 • eecs $\Rightarrow$ K_3

K_3 • Alice-Smith $\Rightarrow$ K_A

naturally have groups: $K_{\bullet}$ friends $\Rightarrow$ K_1 • Alice

$K_{\bullet}$ friends $\Rightarrow$ K_1 • Bob

$K_{\bullet}$ friends $\Rightarrow$ K_1 • mit • eecs • student

Can put group name on ACL

K. friends may read this directory

Given a set of certs, it is possible to tell if some local name (from ACL) can evaluate to your key.

Auth cert: key + right ⇒ key (delegatable, or not)
⇒ name

example: K [read d:/] ⇒ K_1. Alice (no delegation)
right

working group
secure browser
∃ good alg for determining authorization

Certificate revocation

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

fairly high "churn rate"...

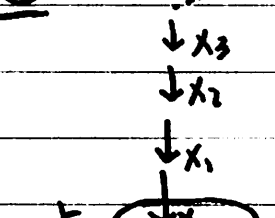
Certificate says "good until 2015-12-01" –
who decides if that is good enough
issuer?
relying party? ⇐ ✓ should be relying party...

issuer has authoritative DB, certs are merely snapshots... [Note that DB itself may not yet fully reflect key compromise, etc...]

method ①: on-line check
ask issuer if cert still good; signed response
OCSP (online certificate status protocol)
heavy load on server!

method ②: CRL's (certificate revocation list)
Server periodically issues CRL, giving list of cancelled (revoked) cert serial #'s, signed
Can get long!

method ③: (Micali)



cert contains end point x_0 of chain of hash values

one day d+i, where d=cert date need x_i (or x_j for $j>i$) to validate cert. Server can give x_i to principal who cert. is about, or issue x_i in response to inquiry...

relying party hashes i times & checks

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

Identity-based encryption (IBE)

everyone has "identity" (e.g. email address)

single TTP T

anyone can encrypt to other, knowing only PK of T & ID of recipient

i.e. Alices pk $\approx (PK_T, \text{"alice@abc.com"})$

Alice's SK – she gets this from T (note trust issue)

uses bilinear maps:

G_1, G_2 groups of prime order q, g generates G_1 (public)

$e: G_1 \times G_1 \to G_2$ s.t. $e(g^a, g^b) = e(g,g)^{ab}$

s = T's SK

g^s = T's PK

H: ID's $\to$ elts of G_1 public hash fn

user gives ID_A to T, gets $\underbrace{H(ID_A)^s}_{\text{Alice's SK}}$ back

To encrypt to Alice: use key $e(H(ID_A), g^s)$

Alice decrypts using key $e(H(ID_A)^s, g)$

$=$ by magic of bilinear maps

- mention LES (Adida/Hohenberger/Rivest)