SDSI -- A Simple Distributed Security Infrastructure

by Ronald L. Rivest
MIT Lab for Computer Science
(joint work with Butler Lampson)

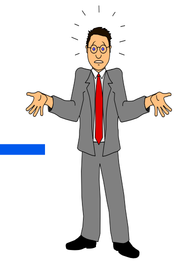
Outline

- Context and history
- Motivation and goals
- SDSI:
  - syntax
  - public keys (principals)
  - naming and certificates
  - groups and access control

The Context

- Public-key cryptography was invented in 1976 by Diffie, Hellman, and Merkle.
- Public-key crypto enables:
  - *Digital signatures* (sign with private key, verify with public key)
  - *Privacy* (encrypt with public key, decrypt with private key)
- But: *Are you using the "right" public key?*

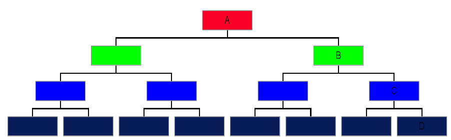
How to Obtain the "Right" PK?



- Directly from its owner
- Indirectly, in a signed message from a trusted CA (*certification agent*):
  - A *certificate* (Kohnfelder, 1978) is a digitally signed message from the CA binding a public key to a name, e.g..:
    "The public key of *Alice B. Smith* is *432102571376553422078986 7* "
  - Certificates can be passed around, or managed in directories.

Scaling-Up Problems

- What if I don't know the CA's public-key?
- How can everyone have a *unique name*?
- "Solution": (PEM, X.509) Use a global hierarchy with one (or few) top-level roots:



- Use *certificate chains* (root to leaf)

Scaling-Up Problems (continued)

- Global name spaces are politically and technically difficult to implement. Legal issues arise if one wants to use certificates to support commerce or legally binding contracts. Standards of due care for issuing certificates must be created.
- A global hierarchical PK infrastructure is slowly beginning to appear (e.g. VeriSign).

Is There a Better Way?

- Reconsider goals...
- "Standard" problems to be solved:
  - Given a public key, identify its owner
  - Find public key for a given party
- "Real" problem to be solved:
  - **build secure distributed computing systems**
    - » *Access control* is paradigmatic application: should a digitally signed request (e.g. http request for a Web page) be honored?

Motivations for designing SDSI:

- Incredibly slow development of PK infrastructure
- Sense that existing PK infrastructure proposals are
  - too complex (ASN.1 encodings, for example)
  - an inadequate foundation for developing secure distributed systems
- A sensed need within W3C security working group for a better PK infrastructure

Related Work

- IETF’s “SPKI” (Simple Public Key Infrastructure) working group (esp. Carl Ellison’s work)
- Blaze, Feigenbaum, and Lacy’s work on “decentralized trust management”
- W3C (world wide web consortium) work on security and on PICS

SDSI Has Very Simple Syntax

- Based on *S-expressions*
- Each S-expression is either:
  - a representation of an octet (byte) string:

```
abc  "Bob Smith"  #4A5B70
=TRa5  #03:def
[unicode] #3415AB8C
```

  - a parenthesized list of simpler S-expressions:

```
( RSA-with-MD5:
  ( E: 3 )
  ( N: #42379F3A0721BB17 ) )
```

Keys are ``Principals''

- In SDSI, the active agents (principals) are *keys*: specifically, the private keys that can make signed statements. We identify a principal with the corresponding verification (public) key:

```
( Principal:
  ( Public-Key:
    ( RSA-with-MD5:
      ( E: #03 )
      ( N: #34FBA341FF73 ) ) )
  ( Principal-At: "http://abc.def.com/" ) )
```

All Keys are Equal*

- Each SDSI principal can make signed statements, just like any other principal.
- These signed statements may be certificates, requests, or arbitrary S-expressions.
- This egalitarian design facilitates rapid “bottom-up” deployment of SDSI.
- * Some SDSI principals may have a special syntax, e.g.: `VeriSign!!  USPS!!`

Signed Objects

- Signing adds a new signature element to end of list representing object being signed.
- A signature can be managed independently of the corresponding signed object.
- An object may be multiply-signed.
- A signature element may itself be signed.

Naming in SDSI

- All names are *local* to some principal.
- A principal can use *arbitrary* local names.
- A principal can *export* name/value bindings by issuing corresponding certificates.
- SDSI syntax supports indirection:

  I can refer to keys (values) named:

```
bob
bob’s alice
bob’s alice’s mother
```

DNS names get special treatment

- A name of the form:

  ```
  billg@microsoft.com
  ```

  is equivalent to:

  ```
  DNS!!'s com's microsoft's billg
  ```

- (This assumes that public keys for entities in the DNS have been created, which may happen in the not too distant future.)

Certificates

- Certificates are signed statements (signed S-expressions).
- Certificates may bind names to values (e.g. to principals or group definitions), may describe the owner of public key, or serve other functions.
- A certificate has an issuer (signer) and an expiration date.

Sample Certificate

```
( Cert:
  ( Local-Name: “John Smith” )
  ( Value: ( Principal: ... ) )
  ( Signed:
    ( Object-Hash: ( SHA-1 #34FD4A ) )
    ( Date: 1996-03-19T07:00 )
    ( Expiration-Date: 2000-01-01T00:00 )
    ( Signer: ( Principal: ... ) )
    ( Signature: #57ACD1 ) ) )
```

Auto-Certificates

- An auto-certificate is signed by the principal whom it is about.
- ```
  ( Auto-Cert:
    ( Public-Key: ... )
    ( Principal-At: ... )
    ( Server: ... )
    ( Name: "Alice B. Cummings" )
    ( Postal-Address: ... )
    ( Phone: ... )
    ( Photo: [image/gif] ... )
    ( Email: alice@abc.com )
    ( Signed: ... ) )
  ```
  ```

On-line orientation

- SDSI assumes that each principal can provide on-line service, either directly or (more typically) indirectly through a server.
- A SDSI server provides:
  - access to a database of certificates issued by the principal
  - access to other objects owned by principal
  - *reconfirmation* service for expired certificates (SDSI does not have CRL's !)

A Simple Query to Server

- A server can be queried:
  "What is the current definition your principal gives to the local name `bob' ?"
- Server replies with:
  – Most recent certificate defining that name, or
  – A signed reply indicating that there is no such definition.

Reconfirmation of Certificates

- SDSI certificates have an expiration date, and may have a *reconfirmation period.*
- A certificate is valid before the expiration date, if the most recent signature is within the last reconfirmation period.
- A principal may authorize its server to reconfirm its certificates.
- Reconfirmation is done by supplying a fresh *reconfirmation signature* to the certificate.

Access Control for WWW Pages

- Motivating application for design of SDSI
- Discretionary access control: server maintains an access-control list (ACL) for each object (e.g. WWW page) managed.
- A central question: how to make ACL’s easy to create, understand, and maintain? (If it’s not easy, it won’t happen.)
- Solution: <u>named groups of principals</u>

Groups

- Distributed version of UNIX "user groups"
- A principal may define a local name to refer to a group of principals:
  - using names of other principals:

    ```
    friends  = ( Group: bob alice tom )
    ```

  - using names of other groups, and algebra:

    ```
    enemies = ( Group:
                  (OR: accountants mgrs ) )
    ```

- Such definitions are given in certificates issued by the defining principal.

Your definitions can use mine

- If you have defined `ron` to refer to my principal (public key), then you can use

```
ron's bob
ron's friends
ron's bob's alice
```

to refer to principals or groups indirectly. (The syntax shown is sugar for things like `( ref: ron bob alice )`)

Sample ACL's

```
( ACL: ( read: associates ) )
( ACL: ( read: Newsweek's subscribers ) )
( ACL: ( read: VeriSign!!'s adults ) )
( ACL: ( read: microsoft's employees ) )
( ACL: ( write: ( OR: bob bob's asst )))
( ACL: ( read:
          ( OR: bob
                bob's friends
                mit's eecs's faculty ) ) )
        ( write: ron ) )
```

Querying for protected objects

- Can make a query for the object.
- If query fails, reply *may* indicate what the (relevant portion of the) ACL is.
- If ACL depends upon remotely-defined groups, *requestor* is responsible for obtaining appropriate ``membership certificate'' and including that as a credential in his query.

Membership Certificates

- Issued by principal defining group, or his server, when requested.
- ```
  ( Membership.Cert:
      ( Member: ( Principal: ... ) )
      ( Group: fudge-lovers )
      ( Signed: ... ) )
  ```
  ```

Encrypted Objects

- ```
  ( Encrypted:
      ( Key:
          ( Key-Hash:
              ( SHA-1 #DA3710 ) ) )
      ( Ciphertext:
          =AZrGT57+30vB1QsMPuI5Ol79 ) )
  ```
- There are a variety of ways to indicate the key:
  - by its hash value
  - in encrypted form
  - through its name
  ```

Other issues and topics

- Multiply-signed requests
- Data compression
- Delegation certificates
- Generalized queries and templates
- Algorithm for evaluating names
- Algorithm for determining group membership

Implementations

- Microsoft (Wei Dai)
- MIT (Matt Fredette)
- We expect working code by end of this calendar year.

To find out more about SDSI

- Draft of our working paper available at:

  `http://theory.lcs.mit.edu/~rivest`

(Warning: under development)

Conclusions

- We have presented a simple yet powerful framework for managing security in a distributed environment.
- Comments appreciated!