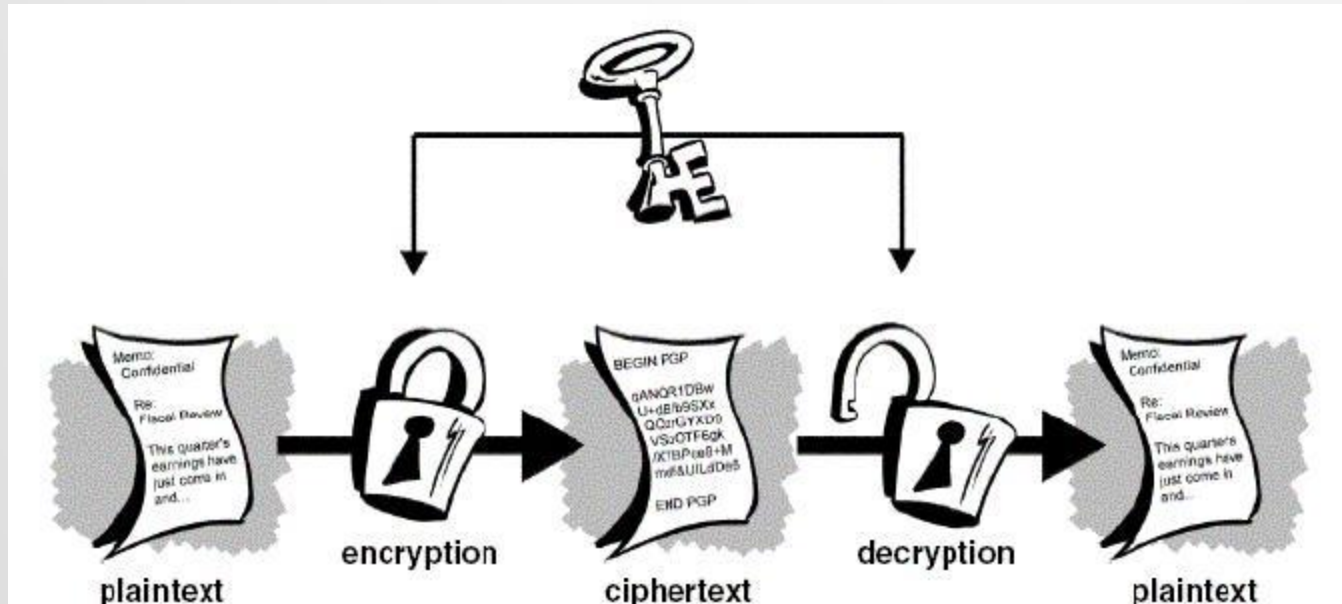


SDSI – A Simple Distributed Security Infrastructure



Mohinish Vinnakota

CS 6750 Fall 2009

Cryptography and Communication Security

Overview

- Principals are public keys

```
( Public-Key:  
  ( RSA-with-SHA1:  
    ( N: =Gt802Tbz9HKm067= )  
    ( E: &11 ) ) )
```

- Egalitarian design-no global hierarchy necessary
- Each principal is a “**certification authority.**”
- Local name spaces
- Simple data structures
- Flexible Signatures
- Identity certificates have human readable content

Overview

- Certificates also give name/value bindings and assert membership.
- On-line Internet orientation.
- Linked local name spaces.

(ref: bob alice) or (ref : <principal> alice)

- Accommodation for “**standard roots**” and global name spaces.

VeriSign!!
IAPR!!
USPS!!
DNS!!

- DNS (Internet email) names have a special status.

Bob.Smith@penguin.microsoft.com
is equivalent to
DNS!!'s com's microsoft's penguin's Bob.Smith

Overview

- A SDSI **group** is typically a set of principals.

friends
mit's biology-department's faculty
(Group: Tom Sam "Bill Gates")

- Clean support for roles.
- Delegation Certificates.

Standard SDSI Object Types

Keys and encryption parameters

Cryptographic keys are represented by an attribute/value object

```
( Public-Key:  
  ( RSA-with-SHA1:  
    ( N: =Hi7KugV013Tv978d00vCpQ== )  
    ( E: &11 ) ) )  
( Private-Key:  
  ( RSA-with-SHA1:  
    ( N: =Hi7KugV013Tv978d00vCpQ== )  
    ( D: &43 ) ) )  
( Shared-Secret-Key:  
  ( RC5-32/12/16-CBC:  
    ( K: "ossifrage" ) ) )
```

Standard SDSI Object Types

Principals as public keys, and their servers

```
( Principal:  
  ( Public-Key: ... )  
  ( Global-Name: ( ref: VeriSign!! WebMaster Bob-Jones ) )  
  ( Principal-At: "http://abc.webmaster.com/cgi-bin/sdsi-server/" )  
  ( Server-At: "http://xyz.webmaster.com/cgi-bin/sdsi-server/" )  
)
```

Standard SDSI Object Types

Encrypted objects

Giving it explicitly in a Key: (attribute/value) field:

(Encrypted:

(Key: (Shared-Secret-Key: ...))

(Ciphertext: =Yh87oKlqpBv8iY55+n== ...))

Giving its hash in a Key-Hash: (attribute/value) field:

(Encrypted:

(Key-Hash: (SHA1 &241dc...))

(Ciphertext: =Yh87oKlqpBv8iY55+n== ...))

Representing it explicitly as an encrypted object itself:

(Encrypted:

(Key: (Encrypted:

(Key-Hash: (SHA1 &548...))

(Ciphertext: &765...)))

(Ciphertext: &345...))

Standard SDSI Object Types

Signed Objects

```
( Signed:  
  ( Object-Hash: ( SHA1: =7Yhd0mNcGFE071QTzXsap+q/uhb= ) )  
  ( Date: 1996-02-14T11:46:05.046-0500 )  
  ( Signature: &3421197655f0021cdd8acb21866b ) )
```


Local Names

- Each principal has its own local name-space.
- A name may be represented in one of two ways:
 - ✓ As an octet string that does not begin with any special character.
Example: "abc", mary-sue, tom@nsf.gov, &61 .
 - ✓ As an arbitrary S-expression n, enclosed in the form
(Local-Name: n).
Example: (Local-Name: (Accounting (Bob Smith)))

Name/Value Bindings

- The principal may assign a value to a local name by issuing a corresponding certificate.
- The binding can be ``symbolic''

“**bob** can bind his local name **lawyer** to **ted's lawyer**”

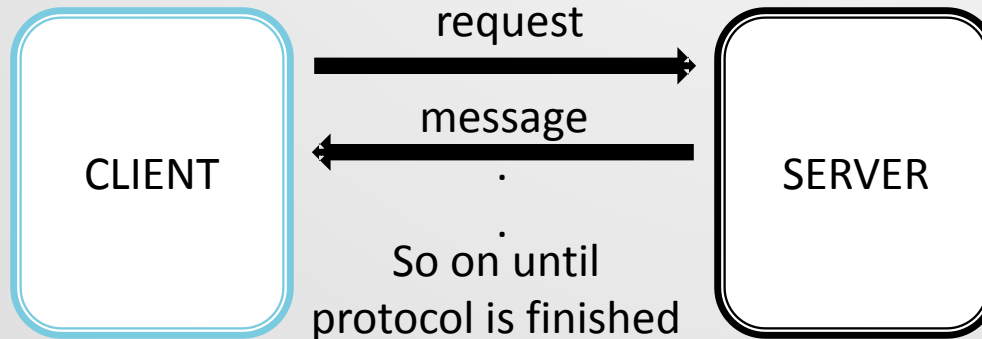
Certificates

- Certificates (certs) are signed (set-type) objects.
- Signed messages are a special case of certificates.

```
( Cert:  
  ( Local-Name: FudgeCo-employees )  
  ( Value: ( Group: "Bill Sweet" "Candy Tooth" "Ima Nut" ) )  
  ( Description:  
    "All current hourly and exempt employees including  
    those on medical or parental leave." )  
  ( ACL: ( read: FudgeCo-management ) )  
  ( Signed: ... ) )
```

Protocols

- ✓ Communication in SDSI takes place as a sequence of protocols between two parties.
- ✓ One party called “Client” and other “Server”.



- ✓ Message can be sent in compressed form.
- ✓ When received it can be decompressed before further processing.
- ✓ If it is of type Encrypted:, the recipient decrypts the message.

Protocols : Queries with “Get” protocol

- Server holds a database of certificates.
- It can be queried to return collections of certificates that satisfies some criteria.
- The **Get** query always contains a `To:` field specifying a principal.
- It specifies a “template” for the desired certificates, giving the object type of desired certificates.

```
(Get.Query:  
  ( To : ( Principal : ... ) )  
  ( Template : ( Cert: ( Local-Name : jim ) ) )  
  ( Signed : ... ) )
```

```
(Get.Reply:  
  ( Your-Last-Message-Hash : ( SHA1 : =tGi0= )  
  ( Reply :  
    ( Cert : ... )  
    ( Cert : ... )  
    ... )  
  ( Signed : ... ) )
```

Protocols : Queries with “Get” protocol

- Server holds a database of certificates.
- It can be queried to return collections of certificates that satisfies some criteria.
- The **Get** query always contains a `To`: field specifying a principal.
- It specifies a “template” for the desired certificates, giving the object type of desired certificates.

```
(Get.Query:  
( To : ( Principal : ... ) )  
( Template : ( Cert: ( Local-Name : jim ) ) )  
( Signed : ... ) )
```

```
(Get.Error :  
( Your-Last-Message-Hash : ( SHA1 : =tGi0= )  
( Error : ... )  
( Signed : ... ) )
```

Protocols : Reconfirmation Queries

- SDSI does not have “certificate-revocation lists.
- Signatures may be designed as needing periodic reconfirmation.

```
( Reconfirm.Query:  
  ( To: ( Principal: ... ) )  
  ( Signed-Object:  
    ( Signed:  
      ( Object-Hash: ( SHA1: &5128 ) )  
      ( Date: 1999-12-25-08:00.000-0500 )  
      ( Signature: &333111 ) ) ) )
```

```
( Reconfirm.Reply:  
  ( Your-Last-Message-Hash:  
    ( SHA1: =Ac8wE1...= ) )  
  ( Signed-Object:  
    ( Signed:  
      ( Object-Hash: ( SHA1: &5128 ) )  
      ( Date: 1999-12-25-08:00.000-0500 )  
      ( Signature: &333111 ) ) )  
    ( Signed:  
      ( Object-Hash: ( SHA1: &a783b0 ) )  
      ( Date: 2000-01-25-12:10.000-0500 )  
      ( Signature: &86723 ) ) ) )
```

Protocols : Reconfirmation Queries

- SDSI does not have “certificate-revocation lists.
- Signatures may be designed as needing periodic reconfirmation.

```
( Reconfirm.Query:  
  ( To: ( Principal: ... ) )  
  ( Signed-Object:  
    ( Signed:  
      ( Object-Hash: ( SHA1: &5128 ) )  
      ( Date: 1999-12-25-08:00.000-0500 )  
      ( Signature: &333111 ) ) ) )
```

```
( Reconfirm.Reply:  
  ( Your-Last-Message-Hash:  
    ( SHA1: =Ac8wE1... ) )  
  ( Failure: <reason> )  
  ( Signed: ... ) )
```


Protocols : Auto-Certs

- An auto-certificate is a special kind of certificate.
- It is distinguished by having been signed by the principal whom it is about.

```
( Auto-Cert:  
  ( Public-Key: ... )  
  ( Principal-At: ... )  
  ( Server: ... )  
  ( Local-Name: ... )  
  ( Global-Name: VeriSign!!'s Wonderland's "Alice McNamee")  
  ( Name: [charset=unicode-1-1] &764511fcc... )  
  ( Description: ... )  
  ( Encryption-Key: ( Public-Key: ... ) )  
  ( Postal-Address: ... )  
  ( Phone: ... )  
  ( Fax: ... )  
  ( Photo: [image/gif] =Yu7gj9D+zX2C... )  
  ( VeriSign-Cert: [application/X.509v3] =GvC492Sq... )  
  ( Email-address: AliceMcNamee@wonderland.com )  
  ( Signed: ... ) )
```

Protocols : Delegation Certificates

The Delegation-Cert: is used to authorize a group (of servers) to be able to sign on behalf of the signing principal.

```
( Delegation-Cert:  
  ( Template: <form> )  
  ( Group: <group> )  
  ( Signed: ... ) )
```

For an example:

```
( Delegation-Cert:  
  ( Template: ( Membership.Cert: ( Group: fudge-lovers ) ) )  
  ( Group: ( Principal: ... (A) ... ) )  
  ( Signed: ... ) )
```

Groups

Groups can be defined by listing their members in a sequence-type object of type `Group`:

For example:

```
( Group: tom mary bill ( Principal : ... ) )
```

Groups can also be defined recursively in terms of other groups:

```
( Group: ( AND: friends over-18 jocks ) ) – intersection
( Group: ( OR: faculty staff friends ) ) – union
( Group: ( NOT: staff ) ) -- ALL! – group
( Group: ( MINUS: staff friends ) ) -- staff that are not friends
( Group: ( ANY: 2 wizards honchos bigwigs ) ) -- threshold of >= 2
( Group: ( OR:
    "Mary Smith"
    friends
    mit's faculty
    ( ref: &32 )
    ( Principal: ... ) ) ) )
( Group: ( OR: alpha ( AND: beta gamma ) ( NOT: delta ) ) )
```

Groups : Membership Queries

- Membership queries are used to obtain membership certificates
- An individual can query a server to ask whether he is a member of a particular group.
- The server can respond with a membership certificate.
 - For very large groups, it may be too expensive to return the entire group definition to a client.

Request

```
( Membership.Query:  
  ( To: ( Principal: ... A ... ) )  
  ( Member: ( Principal: ... B ... ) ... )  
  ( Group: fudge-lovers )  
  ( Credentials: ... )  
  ( Signed: ... ) )
```

Reply

```
( Membership.Cert:  
  ( Member: ( Principal: ... B ... ) ... )  
  ( Group: fudge-lovers )  
  ( Reply: <answer> )  
  ( Hint: <hint> )  
  ( Signed: ... ) )
```

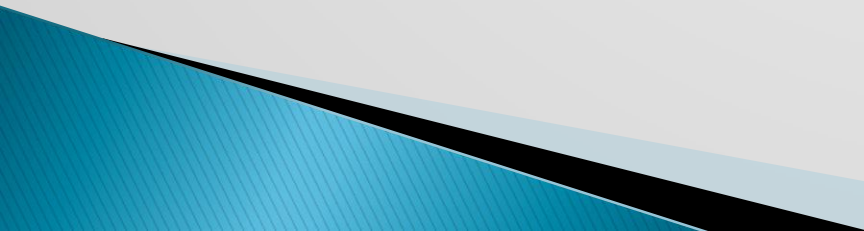
Access-Control Lists

- A group definition have an ACL so that only certain principals may read the definition.
- An ACL is a sequence of the form (ACL: (type1 def1) (type2 def2) ...)
- where each type determines the set of operations being controlled (e.g. read)
- where def is either the local name of a group

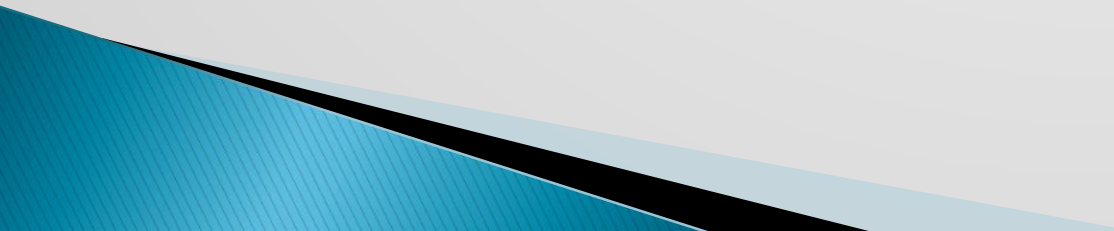
As an example, the certificate for group-23 can only be read by its members:

```
( Cert:  
  ( Local-Name: group-23 )  
  ( Value: ( Group: friends colleagues ) )  
  ( ACL: ( read: group-23 ) )  
  ( Signed: ... ) )
```

Application Scenarios

- Mail Reader
 - World-Wide Web
 - Kerberos-like tickets
 - Distributed signed code
 - Corporate database access
 - Access to medical records
 - Shared-secret key establishment
 - Multi-Cast
- 

Conclusions

- SDSI is a simple yet powerful framework for managing security in a distributed environment.
 - The perspectives and style shown here may assist others in building more secure systems.
- 

References

- 1 Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized trust management. In *Proceedings 1996 IEEE Symposium on Security and Privacy*, page (to appear), May 1996.
- 2 D. Clark and D. Wilson. A comparison of commercial and military computer security policies. In *IEEE Security Privacy*, 1987.
- 3 3rd Donald E. Eastlake and Charles W. Kaufman. *Domain Name System Security Extensions*. Internet DNS Security Working Group, January 30, 1996. (Available at:<ftp://ftp.isi.edu/draft-ietf-dnssec-secext-09.txt>).
- 4 Stephen T. Kent. Internet privacy enhanced mail. *Communications of the ACM*, 36(8):48-60, August 1993.
- 5 National Bureau of Standards. Secure hash standard. Technical Report FIPS Publication 180, National Bureau of Standards, 1993.
- 6 Ronald L. Rivest. The MD5 message-digest algorithm. Internet Request for Comments, April 1992. RFC 1321.
- 7 Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38-47, February 1996.