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

- 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" )
  )
)```
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/" )
)
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: ( Encrypted:
      ( Key-Hash: ( SHA1 &548... ) )
      ( Ciphertext: &765... ) )
    ( Ciphertext: &345... ) )
  ( Ciphertext: &345... ) ) )
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 (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: ... ) )
```
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.
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 : ... ) ) )
Server holds a database of certificates.

It can be queried to return collections of certificates that satisfy 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 : ... ) )

Protopcals : Queries with “Get” protocol
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] =Yu7gi9D+zX2C... )
  ( VeriSign-Cert: [application/X.509v3] =GvC492Sq... )
  ( Email-address: AliceMcNamee@wonderland.com )
  ( Signed: ... ) )
```
The Delegation-Cert: is used to authorize a group (of servers) to be able to sign on behalf of the signing principal.

For an example:

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

( Delegation-Cert:  
  ( Template: ( Membership.Cert: ( Group: fudge-lovers ) ) )  
  ( Group: ( Principal: ... (A) ... ) )  
  ( Signed: ... ) )
Groups can be defined by listing their members in a sequence-type object of type \texttt{Group:}.

For example:

\begin{verbatim}
( Group: tom mary bill ( Principal: ... ) )
\end{verbatim}

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

\begin{verbatim}
( Group: ( AND: friends over-18 jocks ) ) -- intersection
( Group: ( OR: faculty staff friends ) ) -- union
( Group: ( NOT: staff ) ) -- \texttt{ALL!} -- group
( Group: ( MINUS: staff friends ) ) -- staff that are not friends
( Group: ( ANY: 2 wizards honchos bigwigs ) ) -- threshold of $\geq 2$
( Group: ( OR: "Mary Smith" friends mit's faculty ( ref: &32 )
( Principal: ... ) ))
( Group: ( OR: alpha ( AND: beta gamma ) ( NOT: delta ) ) )
\end{verbatim}
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