Lecture 8

Key Management:

Management using all the laws of nature

Wholeness Statement

The key is the secret, not the cryptosystem. Secret keys need to be transferred securely. The best knowledge is reliable knowledge, knowledge is most reliable in the seventh state of consciousness.

Overview

1. Interchange and session keys.
2. Trusted third party creates session keys
3. Sender requests session key from trusted third party, and sends it to recipient.
   - Session keys must be encrypted on the network!
4. Needham-Schroeder Protocol
5. Denning-Sacco Protocol
6. Kerberos

Key Management

Key Management

1. the distribution of cryptographic keys
2. the mechanisms used to bind an identity to a key
3. the generation, maintenance and revoking of such keys

Notation

- X --> Y:{Z}k means that entity X sends entity Y a message Z enciphered with key k
- k_{Bob} refers to a key belonging to Bob
- $k_{Alice,Bob}$ means that Alice and Bob share a key
- If we are talking about a classical cryptosystem k is used to denote a key
- If we are talking about a public key cryptosystem e is used to denote a public key and d is used to denote the private key

Session and Interchange Keys

Definition 9-1 An *interchange key* is a cryptographic key associated with a principal to a communication. A *session key* is a cryptographic key associated with the communication itself.

- Interchange key
  - used to exchange a shared cryptographic key
    - used to encrypt the session key (classical cryptography)
  - used by the receiver to identify the sender
  - used for multiple key exchanges (non-changing)
  - initially acquired in person or some other secure way
    - can later be changed over a secure link (like the password)
- Session Key
  - used for communication between a specific subject and object (e.g., Alice and Bob)
  - different for each communication session (changing)

Key Exchange

The goal of key exchange is to enable Alice to communicate secretly to Bob and vice versa using a shared cryptographic key

Key exchange needs to meet the following criteria:

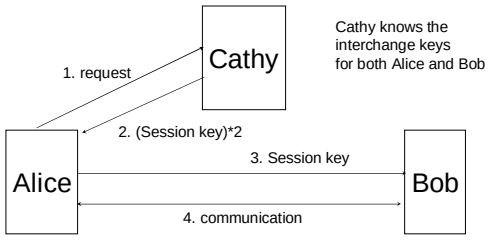
1. An attacker should not be able to eavesdrop and learn the key
2. Alice and Bob may decide to trust a third party (called "Cathy" here)
3. Only the keys are secret. The cryptosystems and protocols are assumed to be known.

Main Point

1. An interchange key is "non-changing"; a session key is changing. The absolute field of pure consciousness is the basis of the everchanging relative field.

Classical Cryptographic Key Exchange and Authentication

Classical Crytographic Key Exchange and Authentication



Cathy encrypts the session key twice, one for Alice and one for Bob.
Alice sends Bob's encrypted session key to Bob when ready to begin communicating.

Classical Cryptographic Key Exchange and Authentication

- First Alice and Bob subscribe to Cathy's service and receive k_{Alice} and k_{Bob}. Only Alice and Cathy know k_{Alice}. Only Bob and Cathy know k_{Bob}.

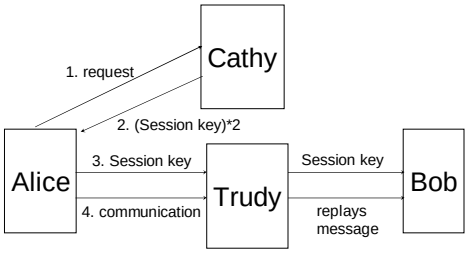
The following is the normal communication protocol.

1. Alice --> Cathy: { request for session key to Bob }k_{Alice}
2. Cathy --> Alice: { $k_{session}$}k_{Alice} || {$k_{session}$}k_{Bob}
3. Alice-->Bob: { $k_{session}$ }k_{Bob}
4. Alice-->Bob: { Transfer $500 dollars to Dan }$k_{session}$

Description of “Normal Protocol”

- Cathy generates a session key, $k_{session}$ that Alice can use in her communication with Bob. Cathy sends back $k_{session}$ encrypted with Alice's key (which she can decipher) and also sends back $k_{session}$ encrypted with Bob's key. Since Alice doesn't know k_{Bob}, she can't decipher it but that doesn't matter since she is just going to forward it to Bob.
- When Bob receives this he can decipher it using k_{Bob} which is a key he shares with Cathy
- Alice then can use $k_{session}$ to communicate securely with Bob.

Replay Attack by Trudy



Trudy intercepts Alice's message
and sends it over and over again.

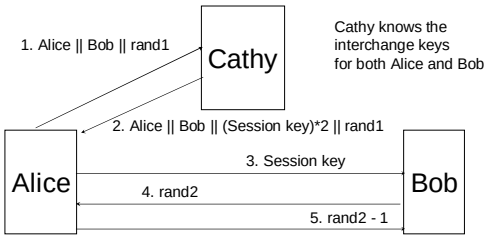
Replay Attack

1. Intercept and record messages 3 and 4 above.
2. Trudy resends { $k_{session}$ }k_{Bob}
   Bob has no idea that Trudy is sending this.
3. Trudy resends
   { Transfer $500 dollars to Dan}$k_{session}$
   Again Bob has no idea that Trudy is sending this. Furthermore, Trudy can replay this message several more times making Alice much poorer.

Needham-Schroeder Protocol

A defense against replay attack
using two random numbers.

Needham-Schroeder Protocol



Alice sends the encrypted session key to Bob.
To authenticate that Alice is indeed the one
sending the key, Bob has Alice modify rand2
in some specific, agreed upon way.

Needham-Schroeder Protocol

1. Alice-->Cathy: { Alice || Bob || rand1 }
2. Cathy-->Alice: { Alice || Bob || rand1 || $k_{session}$ ||
   {Alice || $k_{session}$}k_{Bob}}k_{Alice}
3. Alice-->Bob: { Alice || $k_{session}$ }k_{Bob}
4. Bob-->Alice: { rand2 }$k_{session}$
5. Alice-->Bob: { rand2 - 1 }$k_{session}$
6. Alice-->Bob: { 1. message }$k_{session}$
   Alice-->Bob: { 2. message }$k_{session}$
   Alice-->Bob: { 3. message }$k_{session}$

- The numbers 1, 2, 3 are segment numbers and prevent Trudy from replaying a message.

Description of Needham-Schroeder Protocol

- Alice tells Cathy that she wants to talk to Bob and needs a session key.
- Cathy sends back an identifier of the session (Alice || Bob), the session key, the message encrypted with Bob's secret key, and rand1 (which convinces Alice she is talking to Cathy). This message identifies Alice and has the session key.
- Step 3 is similar to the previous protocol, so Trudy could record this
- But Bob is now checking that it is indeed Alice in steps 4 and 5
- It won't do any good for Trudy to replay step 5 because next time Bob will send a different rand2.
- The segment numbers 1, 2, 3 in step 6 prevent Trudy from replaying a message.

Weakness is Random Session Key

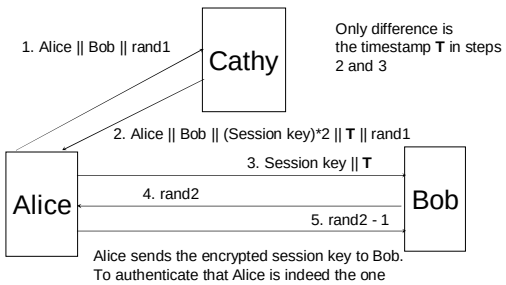
- If session keys are generated with a non-secure random function (one in which it is possible to detect **patterns**), then Trudy could defeat this protocol as follows.

1. Subscribe to Cathy
2. Get several $k_{session}$ keys from Cathy and figure out the algorithm used to generate them so that given $k_{session}$ it is possible to predict what the next $k_{session}$ key is.
3. Get a session key from Cathy
4. Eavesdrop on Cathy to see who the next person is who requests a session key. Assuming it is Alice, Trudy will know what the session key assigned to Alice is. Since Trudy knows the session key she can decipher Bob's random number and respond with the correct enciphered response.

5. Trudy can now replay step 3 of the Needham-Schroeder protocol.
   Trudy-->Bob: { Alice || $k_{session}$ }k_{Bob}
   Note: Trudy does not know what k_{Bob} is but she does know what $k_{session}$ is. Because she doesn't know what k_{Bob} is she is forced to do the following replay
6. Bob-->Alice: { rand3 }$k_{session}$
   Trudy intercepts this. Since she knows $k_{session}$ she will be able to get rand3
7. Trudy-->Bob: { rand3 - 1 }$k_{session}$
   Since Trudy knows $k_{session}$ she can send back the response that Bob is expecting.
8. At this point Bob thinks that Trudy is Alice and since Trudy has $k_{session}$ she can send anything she wants to Bob and Bob will think that Alice is sending it. Furthermore, she can decipher anything that Alice sends during that session.

Denning and Sacco Protocol

Denning and Sacco Protocol



Denning and Sacco Protocol

- Denning and Sacco suggested using timestamps to make the Needham-Schroeder protocol more robust in case the session key is cracked. Their solution enables Bob to detect the replay of step 3.

1. Alice-->Cathy: { Alice || Bob || rand1 }
2. Cathy-->Alice: { Alice || Bob || rand1 || $k_{session}$ || {Alice || **T** || $k_{session}$ }k_{Bob}}k_{Alice}
3. Alice-->Bob: { Alice || **T** || $k_{session}$ }k_{Bob}
4. Bob-->Alice: { rand2 } $k_{session}$
5. Alice-->Bob: { rand2 - 1 } $k_{session}$

- The problem with Denning-Sacco is that it requires that the clocks on the sender and receiver machines be synchronized. This is, in general, difficult to do over the network.

Denning and Sacco Protocol

3. Alice-->Bob: { Alice || T || $k_{session}$ }k_{Bob}

Except for the timestamp T, this is the same as the previous protocol. Now if Trudy records this and plays it back later, Bob will detect that the timestamp T is too old and know that a replay is being attempted.

Otway-Rees Protocol

- Otway-Rees countered with a protocol that doesn't use timestamps. However, we won't look at it since we are more interested in the Denning-Sacco protocol because Kerberos is based on it.
- Uses random numbers in a special way
- End of Chapter 9.2.1

Main Point

2. A protocol is an orderly sequential exchange between two computers. Sequences and orderliness abound in nature. For example, the seasons sequence in an orderly manner.

Kerberos

Disallowing the Birth of An Enemy

To Defend Against Replay Attack

- Needham-Schroeder protocol (1978)
  - Associates a random number with the key exchange
- Denning-Sacco protocol (1981)
  - Adds timestamps to detect replay if random session keys might not really be random
    - (keys cannot have been used in the past or be used in the future by Alice or Bob or be predictable in any way)

Kerberos

- Kerberos uses the Needham-Schroeder protocol as modified by Denning and Sacco.
- Kerberos is used to control access to servers (file server, print server, etc).
- It emphasizes the difference between authentication (determining who the user is) and authorization (determining what the authenticated user is allowed to do).

Kerberos

- Suppose a client, Alice wants to use a server S
- Kerberos requires Alice to use two servers to obtain a credential that will authenticate her to S
  1. Authenticate herself to the Kerberos system
  2. Obtain a ticket to use S
- The name Kerberos refers to the three headed dog from Greek mythology that guarded the gates of hades
- Pity the poor user who tries to get past the three heads of Kerberos!

Kerberos

The three heads are

1. The authentication server that authenticates a user and issues him a ticket that allows him to use the authorization server
2. The authorization (ticket-granting) server checks whether the user is authorized to use the requested server and if so issues him a ticket to use the server.
3. The server checks to make sure the authorization server's ticket is valid and if so grants the user access to the server.

Kerberos

- Composed of a data repository and authentication processes
- The heart of the system is the Key Distribution Center (KDC)
  - Has all the interchange keys of the subjects and objects in the system
  - Composed of the Authentication Server (AS) and the Ticket Granting Server (TGS)
- KDC == Cathy in previous models

Kerberos

- To communicate with Bob
  - Alice must authenticate herself to the system through the Authenticating Server (AS)
  - Alice must obtain a Ticket to communicate with Bob from the Ticket Granting Server (TGS)
    - The ticket contains the session key, timestamp, etc.
  - Bob must make sure he is communicating with Alice and that Alice is approved (through the ticket)

Notation

$k_{X,Y}$ is the **session key** that X and Y will use to communicate.

k_Y is the **interchange key** that server Y and the server that issued the ticket share.

A **ticket** that allows user X to use a server Y looks like this

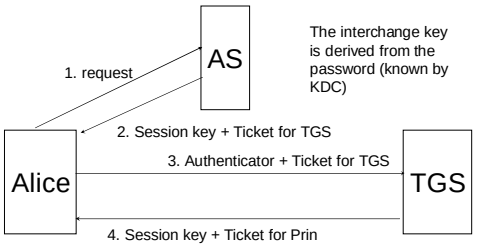
$T_{X,Y} = Y \,||\, \{X \,||\, X \text{ address} \,||\, \text{valid time} \,||\, k_{X,Y}\}k_Y$

An **authenticator** is associated with a ticket and shows that the sender of a ticket is the owner of the ticket.

$A_{X,Y} = \{X \,||\, \text{generation time} \,||\, kt\}k_{X,Y}$

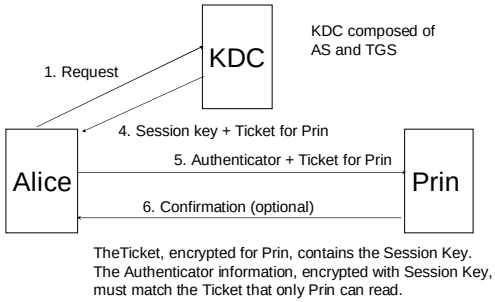
where X is the user, Y is the server and kt is an alternate session key

KDC (Internal Level)



The Session Key from AS allows Alice and the TGS to communicate and is also included in the Ticket from AS

Kerberos (High Level)



TheTicket, encrypted for Prin, contains the Session Key. The Authenticator information, encrypted with Session Key, must match the Ticket that only Prin can read.

Alice wants to use the printer

- The following notations are used in the description of the Kerberos protocol below
  - Auth - the authenticating server
  - Tick - the ticket granting server
  - Prin - the print server
- The following example shows the step by step process required for Alice to access the print server

1. Alice --> Auth : Alice || Tick
   - Alice asks the authenticating server for a ticket to communicate with the ticket granting server.

2. Auth --> Alice : $\{k_{Alice,Tick}\}k_{Alice} \,||\, T_{Alice,Tick}$
   - The authenticating server sends Alice a session key to use when talking to the ticket granting server, and a ticket that lets the ticket granting server know that Alice has been authenticated.
   - The authenticating server knows Alice's password and uses it to generate k_{Alice}. Alice uses the same password to generate k_{Alice}
   - The ticket looks like:

     $T_{Alice,Tick}$
     $= Tick \,||\, \{Alice \,||\, Alice\ address \,||\, valid\ time \,||\, k_{Alice,Tick}\}k_{Tick}$
3. Alice --> Tick : Prin || $A_{Alice,Tick}$ || $T_{Alice,Tick}$
   - Alice requests a ticket to use the print server and sends the ticket received from authenticating server to the ticket granting server along with authenticator to prove that she is the owner of the ticket
   - The authenticator looks like:

     $A_{Alice,Tick} = \{Alice \,||\, generation\ time \,||\, kt\}k_{Alice,Tick}$

4. Tick --> Alice : Alice || $\{k_{Alice,Prin}\}k_{Alice,Tick}$ || $T_{Alice,Prin}$
   - The ticket granting server accesses its database to verify that Alice has permission to use the print server and sends her a session key to use to communicate with the print server and a ticket to give the print server.
   - The ticket looks like:

     $T_{Alice,Prin} = Prin \,||\, \{Alice \,||\, Alice\ address \,||\, valid\ time \,||\, k_{Alice,Prin}\}k_{Prin}$
5. Alice --> Prin : $A_{Alice,Prin}$ || $T_{Alice,Prin}$
   - Alice sends her ticket from the ticket granting server and an authenticator to the print server
   - The authenticator looks like:

     $A_{Alice,Prin} = \{Alice \,||\, generation\ time \,||\, kt\}k_{Alice,Prin}$
6. Prin --> Alice : $\{ t+1 \}k_{Alice,Prin}$
   - Print server convinces Alice that it is really the print server

Main Point

3. A trusted third party is used to exchange the session key in classical cryptography. The trusted third party is, in a seeming paradox, an example of the use of the principle of the second element. Don't remove darkness at the level of darkness, turn on a light (which is not darkness) to remove darkness.

CONNECTING THE PARTS OF KNOWLEDGE WITH THE WHOLENESS OF KNOWLEDGE

1. In classical cryptography it is important to keep the shared key a secret.
2. The Needham-Schroeder protocol is vulnerable to a replay attack if the attacker can figure out a pattern in session keys.

3. Transcendental Consciousness is the secret that TM reveals.

4. Wholeness moving within itself: in Unity Consciousness, the unbounded and bounded are no longer secret; they are known, on the level of direct perception, to be manifestations of one's own Self.