Alin Tomescu, CSE408
Thursday, April 7th, Lecture #19

Key agreement protocols

Needham Schroder

- has been out there for a while until people realized it was broken
- both asymmetric and symmetric key versions
- we have Trent, the trusted party (key distribution center, a.k.a. the KDC)
- Trent has a secret that he shares with Alice k_{TA} and another secret that he shares with Bob k_{TB}
- the description of the protocol assumes the **encryption scheme used is CCA-secure** (no explicit signing)

Protocol

Here's how the protocol works.

Alice (has kta)	**Trent (has kta, ktb)**	**Bob (has ktb)**
$req = \text{alice}, \text{bob}, n_1 \rightarrow$		
	$\leftarrow Enc_{k_{TA}}(n_1, K_s, \text{bob}, F), F = E_{k_{TB}}(K_s, \text{alice})$	
$F = E_{k_{TB}}(K_s, \text{alice}) \rightarrow$		
		$\leftarrow Enc_{k_S}(n_2)$
$Enc_{k_S}(n_2 + 1) \rightarrow$		

Outline:

- Hey Trent, I'm Alice, I'd like to talk to Bob, here's a nonce
  - $req = \text{alice}, \text{bob}, n_1$
  - without tn_1 the attacker with an old session key can impersonate Bob
- Trent will send back to Alice $Enc_{k_{TA}}(n_1, K_s, \text{bob}, F)$, where $F = E_{k_{TB}}(K_s, \text{alice})$
- Alice sends F to Bob
- Bob sends $Enc_{k_S}(n_2)$
- Alice sends $Enc_{k_S}(n_2 + 1)$ (purpose of this is to show that Alice is using the specified session key and is able to decrypt.

Suppose an attacker is able to steal an old session key, and he has the F associated with it then he can impersonate Alice.

To fix this issue, we have to redesign the protocol. A timestamp on F would work decently, making the attack window very small. Another fix is adding more nonces. We are slowly converging to Kerberos.

Fixed protocol

- Alice sends a message to Bob, saying "Hi I'm Alice and I want to talk to you."
  - A, N_A
- Bob sends an encrypted message to Trent, saying "Hi, I'm Bob and Alice wants to talk to me."
  - $B, Enc_{K_B}(A, N_A, R_B), N_B$
- Trent sends to Alice $Enc_{K_A}(B, N_A, K_S, R_B)$ and $Enc_{K_B}(A, K_S, R_B), N_B$
- Alice sends $E_{K_B}(A, K_S, R_B), N_B$ and $E_{K_S}(N_B, R_A)$ to Bob
  - R_B makes sure that K_S is not vulnerable to stolen key attack, so that the K_S is fresh
  - N_B proves that Alice knows K_S

- Is there any proof that Bob knows K_S?
  - Not exactly clear
  - So Alice sends $E_{K_S}(N_B, R_A)$
  - And Bob replies with $E_{K_S}(R_A + 1)$

Kerberos

Kerberos is distributed authentication service. MIT had thousands of students accessing servers from all around the campus so they wanted a single user authentication server for a bunch of different services. The goal was to outsource authentication from the application-specific servers (file server, mail server) to a specialized authentication server.

- users log in with passwords
- users can log in from any workstation
- users don't have to log in all the time

Kerberos is based on the symmetric Needham-Schroeder protocol. It uses a trusted third party, called the *key distribution center (KDC),* consisting of an *Authentication Server (AS)* and a *Ticket Granting Server (TGS)*. Kerberos uses "tickets" to prove the identity of users. The KDC maintains a database of secret keys. Each client on the network whether it's a WS or a server S shares a secret key known with the KDC. Knowledge of this key serves to prove identity.

In the following descriptions, we have the user, who will type in his name and password through a work station (WS). The user wants to access a server S. We're also going to have a special authentications server (AS).

Kerberos v1

Protocol:

- User types in his name and password through a WS. He wants to access server S.
- The WS goes to the AS and says: $req = ID, password, S$
  - Attack: You can steal the password (sent in clear)
- AS sends the WS a ticket $t = Enc_{k_{server}}(ID, S)$
  - Ticket has no freshness
  - No shared key establishment
- WS goes to S and says $m = ID, Ticket$

Kerberos v2

Kerberos v2 has two separate servers for handling authentication:

- main authentication server (AS)
- a ticketing granting server (TGS)

Protocol:

- User enters ID and password into WS
- User sends a message to the AS saying: "Hi, I'm user with ID and I want to talk to TGS for a ticket."
  - $m = ID, TGS$
- AS responds sending $Enc_{k_{ID}}(ticket_1)$
  - AS has a table mapping user ID's to the hash of their passwords
  - $k_{ID} = hash(password\ for\ user\ with\ ID)$
  - $ticket_1 = Enc_{k_{TGS}}(ID, TGS, timestamp_1, timelimit_1)$
    - $timelimit =$ lifetime of the ticket

- User gets it, and decrypts it (he can since the key is the hash of his password), retrieving $ticket_1$
- User sends $ID, \text{server}, ticket_1$ to TGS
    - We need a secure channel between WS and TGS
    - We'll use the AS as a KDC between the WS and TGS
    - We then use the TGS as a KDC between the WS and S
- TGS sends back $ticket_2 = Enc_{k_{server}}(ID, S, timestamp_2, timelimit_2)$
- Alice sends $ticket_2$to server: $ID, ticket_2$