Authentication Protocols and Applications

Introduction to Computer Security

week 10

暨南大學
JINAN UNIVERSITY

Authentication on a single PC

- $A \rightarrow S$: Alice
- $S \rightarrow A$: password?
- $A \rightarrow S$: *****
- ...

Challenges for Network Authentication

- There exists no dedicated physical communication channels. That means an adversary can overhear, intercept, and synthesize any message (constrained by the cryptographic methods used).

Challenges for Network Authentication

- There exists no dedicated physical communication channels. That means an adversary can overhear, intercept, and synthesize any message (constrained by the cryptographic methods used).
- An attacker may
  - Eavesdrop
  - Steal and Intercept messages
  - Forge messages
  - Replay messages
  - Impersonate one or more parties (e.g., man-in-the-middle attack)
  - ...

Challenges for Network Authentication

- There exists no dedicated physical communication channels. That means an adversary can overhear, intercept, and synthesize any message (constrained by the cryptographic methods used).
- An attacker may
  - Eavesdrop
  - Steal and Intercept messages
  - Forge messages
  - Replay messages
  - Impersonate one or more parties (e.g., man-in-the-middle attack)
  - ...
- If the previous protocol is used in the network authentication scenario, an attacker may steal the password, and later use the password to log on.

An Improved (One-way) Authentication Protocol

A password may be encrypted (or hashed) by the public key of S

- $A \rightarrow S$: Alice
- $S \rightarrow A$: password?
- $A \rightarrow S$: $\{*****\}_{pk(S)}$

An Improved (One-way) Authentication Protocol

A password may be encrypted (or hashed) by the public key of S

- $A \rightarrow S$: Alice
- $S \rightarrow A$: password?
- $A \rightarrow S$: $\{*****\}_{pk(S)}$

Attacker Eve may intercept $\{password\}_{pk(S)}$, and replay

- $E \rightarrow S$: Alice
- $S \rightarrow E$: password?
- $E \rightarrow S$: $\{*****\}_{pk(S)}$

An Improved (One-way) Authentication Protocol

A random value (nonce) can be used as a challenge to further improve security

- $A \rightarrow S$: Alice
- $S \rightarrow A$: N
- $A \rightarrow S$: $\{*****, N\}_{pk(S)}$

An Improved (One-way) Authentication Protocol

A random value (nonce) can be used as a challenge to further improve security

- $A \rightarrow S$: Alice
- $S \rightarrow A$: N
- $A \rightarrow S$: $\{*****, N\}_{pk(S)}$

Seems Ok so far, as the last message cannot be used in a replay.

An Improved (One-way) Authentication Protocol

A random value (nonce) can be used as a challenge to further improve security

- $A \rightarrow S$: Alice
- $S \rightarrow A : N$
- $A \rightarrow S : \{*****, N\}_{pk(S)}$

Seems Ok so far, as the last message cannot be used in a replay.

However, An attacker

- may steal the session after Alice logs out

An Improved (One-way) Authentication Protocol

A random value (nonce) can be used as a challenge to further improve security

- $A \rightarrow S$: Alice
- $S \rightarrow A$: N
- $A \rightarrow S$: $\{*****, N\}_{pk(S)}$

Seems Ok so far, as the last message cannot be used in a replay.

However, An attacker

- may steal the session after Alice logs out
- If the Alice is convinced to use a wrong certificate (public key), the protocol is vulnerable to Man-in-the-middle attack

An Improved (One-way) Authentication Protocol

A random value (nonce) can be used as a challenge to further improve security

- $A \rightarrow S$: Alice
- $S \rightarrow A : N$
- $A \rightarrow S : \{*****, N\}_{pk(S)}$

Seems Ok so far, as the last message cannot be used in a replay.

However, An attacker

- may steal the session after Alice logs out
- If the Alice is convinced to use a wrong certificate (public key), the protocol is vulnerable to Man-in-the-middle attack (to demonstrate)

A Two-way Authentication Example

Alice (A), Bob (B), K_{AB} is a shared secret between A and B

- Shared secret used as a challenge:
  $A \rightarrow B$: A, N_1
  $B \rightarrow A$: N_2, $\{N_1\}_{K_{AB}}$
  $A \rightarrow B$: $\{N_2\}_{K_{AB}}$

A Two-way Authentication Example

Alice (A), Bob (B), K_{AB} is a shared secret between A and B

- Shared secret used as a challenge:
  $A \rightarrow B$: A, N_1
  $B \rightarrow A$: N_2, $\{N_1\}_{K_{AB}}$
  $A \rightarrow B$: $\{N_2\}_{K_{AB}}$
- If the used symmetric cipher is known, this protocol increases the chance for offline cryptanalysis

A Two-way Authentication Example

Alice (A), Bob (B), K_{AB} is a shared secret between A and B

- Shared secret used as a challenge:
  $A \rightarrow B$: A, N_1
  $B \rightarrow A$: N_2, $\{N_1\}_{K_{AB}}$
  $A \rightarrow B$: $\{N_2\}_{K_{AB}}$
- If the used symmetric cipher is known, this protocol increases the chance for offline cryptanalysis
- A multi-session replay attack
  $E \rightarrow B$: A, N_1 // starting session one
  $B \rightarrow E$: N_2, $\{N_1\}_{K_{AB}}$
  $E \rightarrow B$: A, N_2 // starting session two
  $B \rightarrow E$: N_3, $\{N_2\}_{K_{AB}}$
  $E \rightarrow B$: $\{N_2\}_{K_{AB}}$ // back to session one

A Two-way Authentication Example

Alice (A), Bob (B), K_{AB} is a shared secret between A and B

- Shared secret used as a challenge:
  $A \rightarrow B$: A, N_1
  $B \rightarrow A$: N_2, $\{N_1\}_{K_{AB}}$
  $A \rightarrow B$: $\{N_2\}_{K_{AB}}$
- If the used symmetric cipher is known, this protocol increases the chance for offline cryptanalysis
- A multi-session replay attack
  $E \rightarrow B$: A, N_1 // starting session one
  $B \rightarrow E$: N_2, $\{N_1\}_{K_{AB}}$
  $E \rightarrow B$: A, N_2 // starting session two
  $B \rightarrow E$: N_3, $\{N_2\}_{K_{AB}}$
  $E \rightarrow B$: $\{N_2\}_{K_{AB}}$ // back to session one
- A possible fix?
  $A \rightarrow B$: A, $\{timestamp\}_{K_{AB}}$
  // Bob verifies $|Clock_B - timestamp| < \delta$

Diffie-Hellman Key Exchange Protocol

- A and B agree on a large prime q and generator a, presumably known by everyone else (including attackers)

  A generates random value X_A

  B generates random value X_B
- $A \longrightarrow B : a^{X_A}\ mod\ q$
- $B \longrightarrow A : a^{X_B}\ mod\ q$
- A computes $(a^{X_B}\ modq)^{X_A}\ mod\ q = a^{X_A X_B}\ mod\ q$

  B computes $(a^{X_A}\ modq)^{X_B}\ mod\ q = a^{X_A X_B}\ mod\ q$

Diffie-Hellman Key Exchange Protocol

- A and B agree on a large prime q and generator a, presumably known by everyone else (including attackers)

  A generates random value X_A

  B generates random value X_B
- $A \longrightarrow B : a^{X_A}\ mod\ q$
- $B \longrightarrow A : a^{X_B}\ mod\ q$
- A computes $(a^{X_B}\ modq)^{X_A}\ mod\ q = a^{X_A X_B}\ mod\ q$

  B computes $(a^{X_A}\ modq)^{X_B}\ mod\ q = a^{X_A X_B}\ mod\ q$

The protocol is vlnerable to man-in-the-middle attack without proper authentication

An Attack on the original Diffie-Hellman Key Exchange

Everyone knows the large prime q and the generator a

- The attacker E is able to control the network traffic
- $A \longrightarrow B(E) : a^{X_A} \ mod \ q$
- $A(E) \longrightarrow B : a^{X_E} \ mod \ q$
- $B \longrightarrow A(E) : a^{X_B} \ mod \ q$
- $B(E) \longrightarrow A : a^{X_E} \ mod \ q$

Alice thinks the shared key with Bob is $a^{X_A X_E} \ mod \ q$
Bob thinks the shared key with Alice is $a^{X_B X_E} \ mod \ q$

Needham-Schroeder Key Distribution Protocol

- At the beginning, everyone has a shared key with the authentication server S
- Alice wants to authenticate with Bob

Needham-Schroeder Key Distribution Protocol

- At the beginning, everyone has a shared key with the authentication server S
- Alice wants to authenticate with Bob
- $A \longrightarrow S : A, B, N_a$
- $S \longrightarrow A : \{N_a, B, K_{ab}, \{K_{ab}, A\}_{K_{BS}}\}_{K_{AS}}$

Needham-Schroeder Key Distribution Protocol

- At the beginning, everyone has a shared key with the authentication server S
- Alice wants to authenticate with Bob
- $A \longrightarrow S : A, B, N_a$
- $S \longrightarrow A : \{N_a, B, K_{ab}, \{K_{ab}, A\}_{K_{BS}}\}_{K_{AS}}$
- $A \longrightarrow B : \{K_{ab}, A\}_{K_{BS}}$
- $B \longrightarrow A : \{N_b\}_{K_{ab}}$
- $A \longrightarrow B : \{N_b - 1\}_{K_{ab}}$

K_{ab} is the one time session key to be used between Alice and Bob.

Needham-Schroeder Key Distribution Protocol

- At the beginning, everyone has a shared key with the authentication server S
- Alice wants to authenticate with Bob
- $A \longrightarrow S : A, B, N_a$
- $S \longrightarrow A : \{N_a, B, K_{ab}, \{K_{ab}, A\}_{K_{BS}}\}_{K_{AS}}$
- $A \longrightarrow B : \{K_{ab}, A\}_{K_{BS}}$
- $B \longrightarrow A : \{N_b\}_{K_{ab}}$
- $A \longrightarrow B : \{N_b - 1\}_{K_{ab}}$

K_{ab} is the one time session key to be used between Alice and Bob.

The Needham-Schroeder Key Distribution protocol is the foundation of Kerberos, which is the earliest and one of the widely used authentication protocol over the Internet today.

Kerberos

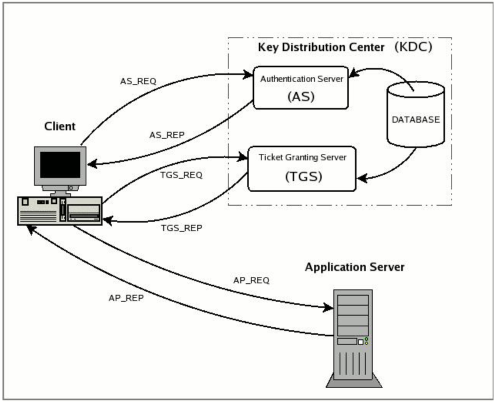
- Part of the project Athena, developed at MIT in 1980s
- Centralized key distribution center (authentication server AS and ticket granting server TGS)
- Exclusively on conventional/symmetric encryption (making no use of public key encryption)
- Versions 1-3 internal at MIT, versions 4 and 5 are widely used today
- Assumes that network connections (rather than servers and workstations) are the weak link in network security: plain password can be eavesdroped, messages can be forged and replayed . . .
- version 4 uses DES as the symmetric cipher, version 5 supports stronger ciphers

Kerberos

- Tries to minimize the number of times a user has to enter password.
  - Not user friendly to type in the password multiple times
  - More exposure of the user's password
- The Authentication Server (AS) in charge of the link between users identities and credentials

  release the burden from other service providers, but all servers in the network have to establish shared secret key with the AS, and **trust** the AS and TGS for authenticating users
- The Ticket Granting Server (TGS) manages the session and redirect users to a variety of application servers
- Authentication procedure should be more automatically done

Kerberos — the Big Picture



The actual protocol is more complex, but we are walking through the essential part (the core) of the actual protocol

Kerberos (without TG Server)

- Initially, Alice wants to access server S
- $A \longrightarrow AS : A, S$
- AS finds the password $passwd(A)$ of A, and generates a random session key k
- $AS \longrightarrow A : \{k, Ticket_{A,S}\}_{passwd(A)}$
  $Ticket_{A,S}$ contains: A, S, t, t_l, k, all encrypted by $k_{AS,S}$
  t is current time, t_l is the life time — for how long the ticket is valid
  A can use $Ticket_{A,S}$ from t to $(t + t_l)$, but cannot decrypt it
- $A \longrightarrow S : Ticket_{A,S}$
  A and S now share the session key k.

Kerberos (without TG Server)

- Initially, Alice wants to access server S
- $A \longrightarrow AS : A, S$
- AS finds the password $passwd(A)$ of A, and generates a random session key k
- $AS \longrightarrow A : \{k, Ticket_{A,S}\}_{passwd(A)}$
  $Ticket_{A,S}$ contains: A, S, t, t_l, k, all encrypted by $k_{AS,S}$
  t is current time, t_l is the life time — for how long the ticket is valid
  A can use $Ticket_{A,S}$ from t to $(t + t_l)$, but cannot decrypt it
- $A \longrightarrow S : Ticket_{A,S}$
  A and S now share the session key k.

- Downside: when Alice wants to access a different server S', she will have to request the AS for $TGT_{A,S'}$ again, and then type in her password again

Kerberos — TGS

AS authenticates a user once, Ticket Granting Server provides a ticket-granting service

Kerberos — TGS

AS authenticates a user once, Ticket Granting Server provides a ticket-granting service

- $A \longrightarrow AS : A$
- AS finds the password $passwd(A)$ of A, and generates a random session key k
- $AS \longrightarrow A : \{k, Ticket_{A,TGS}\}_{passwd(A)}$
  $Ticket_{A,TGS}$ contains: A, TGS, t, t_l, k, all encrypted by $k_{AS,TGS}$
- When Alices asks for service from S
  $A \longrightarrow TGS : A, S, Ticket_{A,TGS}$
- $TGS \longrightarrow A : \{k', Ticket_{A,S}\}_k$
  $Ticket_{A,S}$ contains: A, S, $t+1$, t_l, k', all encrypted by $k_{S,TGS}$

Kerberos — TGS

AS authenticates a user once, Ticket Granting Server provides a ticket-granting service

- $A \longrightarrow AS : A$
- AS finds the password $passwd(A)$ of A, and generates a random session key k
- $AS \longrightarrow A : \{k, Ticket_{A,TGS}\}_{passwd(A)}$
  $Ticket_{A,TGS}$ contains: A, TGS, t, t_l, k, all encrypted by $k_{AS,TGS}$
- When Alices asks for service from S
  $A \longrightarrow TGS : A, S, Ticket_{A,TGS}$
- $TGS \longrightarrow A : \{k', Ticket_{A,S}\}_k$
  $Ticket_{A,S}$ contains: A, S, $t+1$, t_l, k', all encrypted by $k_{S,TGS}$

Next time, if Alice wants to access S', TGS will issue a new $Ticket_{A,S'}$ with a new session key (to be used between A and S'), and increment t again

Kerberos — TGS

In the above protocol

- User does not need to send password over the network (confidentiality for the password is guaranteed with good usability)
- Password management by the Authentication server, all by using a symmetric cipher

Kerberos — TGS

In the above protocol

- User does not need to send password over the network (confidentiality for the password is guaranteed with good usability)
- Password management by the Authentication server, all by using a symmetric cipher
- Ticket granting service by TGS
- Initially password authentication, later on authentication by shared secret keys

Kerberos — TGS

In the above protocol

- User does not need to send password over the network (confidentiality for the password is guaranteed with good usability)
- Password management by the Authentication server, all by using a symmetric cipher
- Ticket granting service by TGS
- Initially password authentication, later on authentication by shared secret keys
- Establish a one time session key that lives a short life time between the user and TGS

Kerberos — TGS

In the above protocol

- User does not need to send password over the network (confidentiality for the password is guaranteed with good usability)
- Password management by the Authentication server, all by using a symmetric cipher
- Ticket granting service by TGS
- Initially password authentication, later on authentication by shared secret keys
- Establish a one time session key that lives a short life time between the user and TGS
- Downside: when the session key expires, a user needs to re-authenticate with *AS*

Kerberos — TGS

The use of an authenticator to extend life time

- $A \longrightarrow TGS : A, S, Ticket_{A,TGS}, Authenticator_{A,TGS}$
- $TGS \longrightarrow A : \{k_{A,S}, t+1, Ticket_{A,S}\}_k$
  k is the one time session key between A and TGS
  $Ticket_{A,S}$ contains: A, S, $t+1$, t_l, k', all encrypted by $k_{S,TGS}$
  $Authenticator_{A,TGS}$ contains: A, $Address_A$, t, encrypted by $k_{A,TGS}$

Kerberos — TGS

The use of an authenticator to extend life time

- $A \longrightarrow TGS : A, S, Ticket_{A,TGS}, Authenticator_{A,TGS}$
- $TGS \longrightarrow A : \{k_{A,S}, t+1, Ticket_{A,S}\}_k$
  k is the one time session key between A and TGS
  $Ticket_{A,S}$ contains: A, S, $t+1$, t_l, k', all encrypted by $k_{S,TGS}$
  $Authenticator_{A,TGS}$ contains: A, $Address_A$, t, encrypted by $k_{A,TGS}$
- $Authenticator_{A,TGS}$ not only extends session life time, but also serves as a way of authentication between A and TGS

Kerberos — TGS

The use of an authenticator to extend life time

- $A \longrightarrow TGS : A, S, Ticket_{A,TGS}, Authenticator_{A,TGS}$
- $TGS \longrightarrow A : \{k_{A,S}, t+1, Ticket_{A,S}\}_k$
  k is the one time session key between A and TGS
  $Ticket_{A,S}$ contains: A, S, $t+1$, t_l, k', all encrypted by $k_{S,TGS}$
  $Authenticator_{A,TGS}$ contains: A, $Address_A$, t, encrypted by $k_{A,TGS}$
- $Authenticator_{A,TGS}$ not only extends session life time, but also serves as a way of authentication between A and TGS
- Another authenticator is used between A and the service provider S for mutual authentication