Authentication / Key Distribution

Mahalingam Ramkumar
Department of CSE
Mississippi State University
So we have a shared secret...

- What next?
- Alice and Bob share a secret K
- How do they determine that they do?
- Challenge-Response Protocols
  - Alice issues a challenge
  - Bob replies
    - Now Alice knows that Bob is authentic
  - Bob issues a challenge
  - Alice responds
    - Now Bob knows Alice is authentic
Two-way Authentication

A

Rb

E(K,Rb)

Ra

E(K,Ra)

Alice

Bob
Two-way Authentication
Simplified

Alice

Bob

A, Ra

Rb, E(K,Ra)

E(K,Rb)
Not so simple! (Reflection Attack)

The trick: Make challenger answer its own question!
Things to bear in mind

- Four general rules
  - Have initiator prove who he/she is before responder has to
  - Initiator and responder should use different keys
  - Initiator and responder should draw challenges from different sets – odd / even
  - Watch out for other parallel sessions
Authentication using HMAC

Alice

Ra

Rb, HMAC(Ra, Rb, A, B, K)

HMAC(Ra, Rb, K)

Bob
Key Establishment
Diffie - Helman

Alice picks a

Bob picks b

Large prime $p$, generator, $g$

$p, g, \alpha \equiv g^a \mod p$

$\beta \equiv g^b \mod p$

$K_{AB} \equiv \beta^a \equiv \alpha^b \equiv g^{ab} \mod p$
Man in the Middle

Alice picks a

Large prime \( p \), generator, \( g \)

Bob picks \( b \)

\[ p, g, \alpha \equiv g^a \mod p \]

\[ \omega \equiv g^w \mod p \]

\[ K_{AB1} \equiv g^{aw} \mod p \]

Oscar picks \( w \)

\[ p, g, \omega \equiv g^w \mod p \]

\[ \beta \equiv g^b \mod p \]

\[ K_{AB2} \equiv g^{bw} \mod p \]

Alice and Bob are not aware of the man-in-the-middle!
Key Establishment
RSA

Alice picks $p_a, q_a$
\[ n_a = p_a q_a, \phi(n_a) = (p_a - 1)(q_a - 1) \]
Alice picks $e_a$.
and $d_a \equiv e_a^{-1} \mod \phi(n_a)$
Destroys $p_a, q_a, \phi(n_a)$
Alice chooses $K_a$

Bob picks $p_b, q_b$
\[ n_b = p_b q_b, \phi(n_b) = (p_b - 1)(q_b - 1) \]
Bob picks $e_b$.
and $d_b \equiv e_b^{-1} \mod \phi(n_b)$
Destroys $p_b, q_b, \phi(n_b)$
Bob chooses $K_b$

\[ K_{AB} = K_a \oplus K_b \]
Key Establishment with Asymmetric Crypto

- Exchange public keys
- Established secret based on public and private keys
- Alice – public key X
- What prevents Oscar from saying X is his public key?
- Somebody has to certify that X is indeed Alice's public key
- Who can do that?
  - How?
- PKI (Public Key Infrastructure)
Certificate Authority

- CA **signs** public keys with the CA's private key
- Everybody has access to CA's *public* key
  - Public knowledge
  - Announced in reputable sources (like newspapers)
  - Preloaded in all computers (or browsers)
- Only CA can sign / revoke certificates
- Anybody can verify a certificate
- X.509 certificate
## X.509 Formats

<table>
<thead>
<tr>
<th>Version</th>
<th>Algorithm/Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cert S.No</td>
<td>Issuer Name</td>
</tr>
<tr>
<td>Algorithm/Parameters</td>
<td>Update date</td>
</tr>
<tr>
<td>Issuer</td>
<td>Next Update</td>
</tr>
<tr>
<td>Validity - Not Before / After</td>
<td>Revoked Certificates</td>
</tr>
<tr>
<td>Subject</td>
<td>.</td>
</tr>
<tr>
<td>Algorithms Parameters</td>
<td>Signature</td>
</tr>
<tr>
<td>Key</td>
<td>Revocation List</td>
</tr>
<tr>
<td>Issuer ID</td>
<td>Certificate</td>
</tr>
<tr>
<td>Subject ID</td>
<td></td>
</tr>
<tr>
<td>Extensions</td>
<td>V1</td>
</tr>
<tr>
<td>Signature</td>
<td>V2</td>
</tr>
<tr>
<td>Certificate</td>
<td>V3</td>
</tr>
</tbody>
</table>
Certificate Authorities

- CA<<X>> - certificate of X issued by CA
- Y<<X>> - certificate of X issued by Y
- Not very practical to have a single global certificate authority
- Issuing certificates is a commercial service
- Verisign, Microsoft, Entrust, beTrusted, eOriginal, SecureNet .....hundreds of issuers!
- If Alice and Bob have different CA's how can they verify each other's certificates?
A and B can verify each other's certificate
C and D can verify each other's certificate

How can A and C verify each other's certificate?
A and B can verify each other's certificate

C and D can verify each other's certificate

How can A and C verify each other's certificate?

Now A/B have authenticated UY, C/D have authenticated UX
For node A/B - X<<T>> authenticates UT, T<<Y>> authenticates UY
For node C/D - Y<<T>> authenticates UT, T<<X>> authenticates UX
T<<X>> - Forward Certificate
X<<T>> - Reverse Certificate

How can A and C verify each other's certificate?
Now A/B have authenticated UY, C/D have authenticated UX
Key establishment without asymmetric crypto

- Basic key distribution
- Kerberos
- Key predistribution
Basic Key Distribution (BKD)

- N nodes
- Each node needs to share a key with every other node
  - Each node needs N-1 keys
- How many unique keys
  - Number of unique pairs – N(N-1)/2
- Does not scale well!
- Very impractical to establish keys
- If a new node is added – a corresponding key should be given to all other nodes.
Kerberos

- Basic purpose
  - Establish authenticated session keys
- Alice and Bob share a secret with a server
- Alice and Bob need to establish a shared secret
- Shared secret establish by mediation with server
- Certificate authority (for PKI) was OFFLINE
- Trusted server needs to be ONLINE
Kerberos (Extremely Simplified)

- KA is key shared between A and server
- A to server
  - \( \{A|B|E_{KA}(A|B)\} \)
- Server to A
  - \( E_{KA}(K|T) \)
  - \( T = E_{KB}(K|A|B) \)
- A to B
  - \( T = E_{KB}(K|A|B) \)
- K is the session key
Kerberos Authentication Service

• Components
  – Workstations (clients) - C
  – Servers (mail server, print server, file server) - V
  – Authentication server - AS
  – Ticket granting server – TGS

• Typical scenario
  – User logs in once every day – enters password
    • Uses different services at different times during the day
    • Totally unaware of Kerberos.

• Kerberos client handles “nasty details” on behalf of the user.

• All clients and TGS(s) share a key with AS
• All servers (V) share a secret with TGS
More details

- Once a day
  - C -> AS : ID_{C}||ID_{TGS}
  - AS -> C: E_{KC}(T_{TGS})
  - T_{TGS} = E_{KTGS}(ID_{C}||AD_{C}||ID_{TGS}||TS_{1}||L_{1})

- Once per type of service
  - C->TGS : ID_{C}||ID_{V}||T_{TGS}
  - TGS->C: T_{V}
  - T_{V} = E_{KV}(ID_{C}||AD_{C}||ID_{V}||TS_{2}||L_{2})

- Once per service session
  - C->V: T_{V}
Multiple Kerberi

- \{\text{AS, TGS, all Cs, all Vs}\} - Kerberos Realm
- Multiple Kerberi – many realms
- Clients may need to access service from other realms
- TGSs of different realms share a unique secret
- If N realms, each TGS stores N-1 additional keys – total N(N-1)/2 such keys
- Client approaches local TGS for a remote TGS ticket.
Key pre-distribution (KPD)

A TA chooses P secrets $\Phi$
Each node has a unique ID
Each node gets $k$ keys
Node A gets $\Phi_A = F(\Phi, A)$
Node B gets $\Phi_B = F(\Phi, B)$
Shared secret between A, B is
$$K_{AB} = G(\Phi_A, B) = G(\Phi_B, A)$$
The function $G$ is public
BKD is also a KPD!
The set of $N(N-1)/2$ keys is $\Phi$
$$\Phi = \{K_{xy}\} \text{ for all } x, y, x \neq y$$
$\Phi_A = F(\Phi, A) = \{K_{Ax}\} \text{ for all } x, x \neq A$
$$G(\Phi_A, B) = K_{AB} = G(\Phi_B, A)$$
Key pre-distribution (KPD)

TA chooses a n-degree symmetric polynomial
\[ F(x, y) \equiv \sum_{i=0}^{n} \sum_{j=0}^{n} a_{ij} x^i y^j \pmod{p}, \quad a_{ij} = a_{ji} \]

Number of unique values of \( a_{ij} \) is \((n+1)^2 / 2\)
The TA's secrets are \( \Phi = \{ a_{ij} \} \)
\[ \Phi_A \equiv G_A(x) \equiv F(x, A) \pmod{p} \]
\[ \Phi_B \equiv G_B(x) \equiv F(x, B) \pmod{p} \]

\( G_A(x), G_B(x) \) are n degree polynomials in one variable
Coefficients of \( G_A(x) \) are the \( n+1 \) secrets \( \Phi_A \)
Coefficients of \( G_B(x) \) are the \( n+1 \) secrets \( \Phi_B \)

\[ K_{AB} \equiv G_A(B) = G_B(B) \equiv F(A, B) \]

The scheme is \( n \)-secure
\( n \) or less nodes, pooling all their secrets cannot break the system
\( n+1 \) nodes can!