Strong Password Protocols
Strong Password Protocols

- Password authentication over a network
  - Transmit password in the clear.
    - Open to password sniffing.
    - Open to impersonation of server.
  - Do Diffie-Hellman exchange to establish a secure key and an encrypted tunnel.
    - Open to impersonation of server.
  - Establish a SSL connection. Use trust anchors for mutual authentication of machines.
    - Security depends on the security of the trust anchors.
Strong Password Protocols

- **Password authentication over a network**
  - Compute a hash of the password. Use the hash as a secret key in a challenge response scheme.
    - **Scheme:**
      - Alice to Bob: Login request
      - Bob to Alice: Here is challenge $R$.
      - Alice to Bob: $f(\text{hash(password)}, R)$
    - Open to dictionary attack by eavesdropper or someone impersonating Bob.
Strong Password Protocols

- Password authentication over a network
  - Use a one-time password
    - Lamport Hash
    - S/Key
  - Use a **strong password protocol**
    - Secure from dictionary attacks by impersonator or eavesdropper.
    - Secure against impersonator on either side.
Lamport Hash

- Bob stores
  - Username Alice
  - int n
  - $h^n(password)$, $h$ – one way function

- Password generation:
  - Alice chooses a password.
  - Alice calculates $h^n(password)$ and sends the hash value and n to Bob.
  - Bob initializes the database entry.
Lamport Hash

Protocol:

Alice

password

Alice’s Workstation

Bob checks:
Is $h(Alice\’s\ answer) = h^n(password)$.
If yes, authenticate.
Then replace $n$ with $n-1$ and store $h^{n-1}(password)$.
To prevent password guessing, randomly generate a salt and store it at the server.

Calculate $h^i(pwd,salt)$
Lamport’s Hash

- Alice’s workstation needs to regenerate the scheme with a **new** password whenever n counts down to 1.
- There is no mutual authentication.
- Vulnerable to the “small n” attack.
Lamport’s Hash

- Mallory impersonates Bob.
- Alice tries to log on to Bob, but talks to Mallory.
- Mallory sends a small $m < n$.
- Alice sends $h^{m-1}(pwd)$
- Mallory calculates $h^{n-1-m+1}(h^{m-1}(pwd))$
- Mallory can now login to real Bob
S/Key

- Deployed version of Lamport hash
- RFC 1938
- http://www.openbsd.org/faq/faq8.html#SKey
Encrypted key exchange

- Alice has a “weak” password pswd.
- Bob stores a hash $W = h(\text{pswd})$ of the password.
- Alice’s workstation knows how to calculate $W$ on the fly, once Alice types in her password.
- Use $W$ in a way that does not give any hints on $W$. 
Alice and Bob share a weak secret $W = h(\text{password})$.

Alice chooses a random number $a$. She sends: $W \{g^a\}$

Bob chooses a random $b$ and a challenge $C_1$. He sends: $W \{g^b, C_1\}$

Both Bob and Alice use their knowledge of $W$ to encrypt their mutual messages. They both calculate $K = g^{ab}$.

Alice then proves her knowledge of $W$ by her ability to calculate $K$. She also picks a challenge $C_2$ and sends $K \{C_1, C_2\}$ to Bob.

Bob encrypts this message and finds that Alice has solved his challenge $C_1$. Finally, Bob authenticates himself to Alice. He proves his knowledge of $W$ by his knowledge of $K$, which he proves by being able to correctly encrypt Alice’s challenge $C_2$. He sends $K \{C_2\}$ to Alice.
**Encrypted key exchange**

### EKE: Diffie-Hellman exchange with encryption.

<table>
<thead>
<tr>
<th>Alice: &quot;Alice&quot;, $E_W(g^a)$</th>
<th>Bob: $E_W(g^b)$, Challenge $C_{Bob}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>At this point, both Alice and Bob calculate $K = g^{ab}$</td>
</tr>
</tbody>
</table>

$E_K(C_{Alice}, C_{Bob})$

$E_K(C_{Alice})$
EKE: Encrypted Key Exchange

- Secure against eavesdropper because all data are undistinguishable from random numbers. Eavesdropper cannot decide whether the $g^a$, $g^b$ are the correct decryption.

- Secure against impersonation:
  - If treacherous Trudy impersonates Bob, she guesses a single value $W$ in the first exchange.

See the definition again.
Encrypted key exchange

**EKE: Diffie-Hellman exchange with encryption.**

W \[
\text{hash of password. (Bob stores it, Alice can recalculate it).}
\]

<table>
<thead>
<tr>
<th>Alice: &quot;Alice&quot;, $E_W(g^a)$</th>
<th>Bob: $E_W(g^b)$, Challenge $C_{Bob}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>At this point, both Alice and Bob calculate $K = g^{ab}$</td>
<td></td>
</tr>
<tr>
<td>$E_K(C_{Alice}, C_{Bob})$</td>
<td>$E_K(C_{Alice})$</td>
</tr>
</tbody>
</table>
**SPEKE: Simple Password Exponential Key Exchange**

- Use W in place of g in the Diffie Hellman exchange.

<table>
<thead>
<tr>
<th>Alice: &quot;Alice&quot;, $W^a$</th>
<th>Bob: $W^b$, Challenge $C_{Bob}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>At this point, both Alice and Bob calculate $K = W^{ab}$</td>
</tr>
<tr>
<td>$E_K(C_{Alice}, C_{Bob})$</td>
<td>$E_K(C_{Alice})$</td>
</tr>
</tbody>
</table>
**PDM: Password Derived Moduli**

- PDM uses modulus $p$ that is a function of the password and uses 2 for $g$ in Diffie Hellman.

<table>
<thead>
<tr>
<th>Alice: &quot;Alice&quot;, $E_W(2^a \mod p)$</th>
<th>Bob: $E_W(2^b \mod p)$, Challenge $C_{Bob}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>At this point, both Alice and Bob calculate $K = 2^{ab}$</td>
<td></td>
</tr>
<tr>
<td>$E_K(C_{Alice}, C_{Bob})$</td>
<td>$E_K(C_{Alice})$</td>
</tr>
</tbody>
</table>
Strong Passwords: EKE

- A bad implementation of EKE allows an eavesdropper to exclude passwords.
  - Assume that we calculate in the field of number modulo $p$, $p$ a prime.
  - Then $g^a$ and $g^b$ are both $m$ bit numbers smaller than $p$.
  - Attacker maintains a dictionary of possible passwords and observes many authentication rounds.
  - If $W$ is in the dictionary, he encrypts Alice’s round 1 message $M$. If $W^{-1}\{M\} > p$, then attacker excludes $W$.
  - Chance of excluding a false password $W$ is $2^m - p / p$.
  - If this chance is about 80%, then 50 rounds determine the password out of a normal dictionary.
Augmented Strong Password Protocols

- If someone knows W in EKE, they can impersonate Alice.
- Augmented Protocols
  - Trudy can steal Bob’s database
  - Trudy can do a dictionary attack.
  - If the dictionary attack is unsuccessful, then she cannot impersonate Alice.
    - Knowledge of W is not enough.
Augmented EKE

- Bob stores “Alice, p, g^w mod p”

<table>
<thead>
<tr>
<th>Alice: &quot;Alice&quot;, g^a mod p</th>
<th>Bob: g^b + g^w mod p, u, Challenge C_{Bob}</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>At this point, both Alice and Bob calculate K = g^{b(a+uW)} mod p</td>
</tr>
<tr>
<td>Alice: E_k(C_{Bob}), C_{Alice}</td>
<td>Bob: E_k(C_{Alice})</td>
</tr>
</tbody>
</table>
Strong Passwords

- Augmented PDM
  - Server Information Creation
    - Alice has password $pssw$
    - Alice sends to Bob
      - $p = f (pssw)$ [this is a prime]
      - $W = hash (pssw)$ [one-way hash]
    - Bob stores:
      - Alice, $p$, $W$,
Strong Passwords: Augmented PDM

Alice creates random number \( a \).
She re-computes \( W \) and \( p \) from her password.

\[
\text{Alice} \quad 2^a \mod p \quad \text{Bob}
\]

Bob chooses a random number \( b \).
Bob calculates \( 2^b \mod p \).
Bob sends \( 2^b, \text{hash}_1(2^{ab} \mod p, 2^{bw} \mod p) \) to Alice

Alice knows that Bob is Bob because Bob proves that he knows \( 2^{bw} \). Alice now sends \( \text{hash}_2(2^{ab} \mod p, 2^{bw} \mod p) \)

Bob knows that Alice is Alice because she proves to him that she knows \( W \). If Alice had just broken into the server, she would have to calculate \( 2^{bw} \) from \( 2^W \mod p \).
Augmented PDM

- Bob stores Alice, \( p \), \( 2^w \mod p \) and picks \( b \)
- Alice computes \( p \) and \( W = \text{hash(passwd)} \) from the password and picks \( a \).

<table>
<thead>
<tr>
<th>Alice: &quot;Alice&quot;, ((2^a \mod p))</th>
<th>Bob: ((2^b), \text{Hash}_1(2^{ab} \mod p, 2^{bw} \mod p))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice: Hash(_2)((2^{ab} \mod p, 2^{bw} \mod p))</td>
<td></td>
</tr>
</tbody>
</table>
Strong Passwords

- Secure Remote Password
  - RFC 2945
  - Bob stores \{Alice, \( g^W \mod p \)\}, where \( W = f(\text{passwd}) \).
Strong Passwords

Secure Remote Password (SRP)

Alice creates random $a$ and sends $g^a$ to Bob.

Bob creates random $b$, challenge $C_{BOB}$ and 32b number $u$.

Bob sends $g^b + g^W \mod p$, $u$, $C_{BOB}$ to Alice.

Both calculate $K = g^{b(a+uW)} \mod p$.

Alice sends $K \{C_{Bob}\}$, $C_{Alice}$ to Bob.

Bob sends $K \{C_{Alice}\}$ to Bob.