Lecture 6

Strong Password Protocols
Using Passwords

- send pwd, compare against h(pwd)
- send h(pwd), compare against h(pwd)
- send h(pwd), compare against h(h(pwd))
- use h(pwd) as secret in challenge/response, server stores h(pwd). Why not h(h(pwd))? 
- Lamport’s hash
- anonymous Diffie-Hellman, then one of above
- server sends cert, establish SA, do one of above
Alternative: Strong password Protocols

- Requires no user-specific info (like trust anchors)
- Someone who impersonates Bob or Alice, or who eavesdrops, cannot gain information for dictionary attack
EKE (Encrypted Key Exchange)

share weak secret $W=h(\text{pwd})$

Alice

choose A

“Alice”, \{g^A \mod p\}W

Bob

choose B, challenge C1

\{g^B \mod p, C1\}W

choose C2

$K=g^{AB} \mod p$

\{C1, C2\}K

\{C2\}K
SPEKE (simple password exponential key exchange) share weak secret $W = h(\text{pwd})$

Alice

choose A

Bob

choose B, challenge $C_1$

---

\[ W^A \mod p \]

---

\[ W^B \mod p, C_1 \]

---

choose $C_2$

\[ K = W^{AB} \mod p \]

\[ \{C_1\}K, C_2 \]

\[ \{C_2\}K \]
PDM (password derived moduli)

compute appropriate prime $p = f(pwda)$

Alice

choose A

“Alice”, $2^A \mod p$

Bob

choose B, challenge $C_1$

$2^B \mod p, C_1$

stores $p$

choose $C_2$

$K = 2^{AB} \mod p$

$\{C_1, C_2\}K$

$C_2$
These depend on getting no information

• original EKE paper had lots of variants, almost all of which were found to be flawed. Tried to base it on other public key schemes besides D-H, only D-H worked.

• But suppose do D-H straightforward way. Eavesdropper sees \( \{g^A \mod p\}^W \). Any info?
Similar SPEKE Vulnerability

- Only half the numbers are squares mod p
- Squares only generate half the numbers
- How tell if something is a square mod p?
- So eavesdropper can eliminate passwords if see $W^x \mod p$, and it isn’t a square
- What is the probability on any exchange?
- Why isn’t this as serious as previous slide?
PDM

- want $p$ to be a “safe prime” ($p-1/2$ also prime)
- if $p=3 \mod 8$, 2 will be a generator (obscure math...law of quadratic reciprocity)
- $p$ has to be 2 mod 3
- so $p$ has to be 11 mod 24
Potential Vulnerability

• If $g^A \mod p$ is greater than the p for some password, then an eavesdropper can eliminate that password

• Fix it by choosing p from a very small space (top 64 bits fixed)

• Also near a power of 2 to make maximal use out of the bits
Another potential PDM Vulnerability

- What happens if Trudy chooses $A$ such that $2^A < p$?
- Can Bob detect this easily? (hint: what’s $g$?)
Password Guessing

• “Humans are incapable of securely storing high-quality cryptographic keys, and they have unacceptable speed and accuracy when performing cryptographic operations. They are also large, expensive to maintain, difficult to manage, and they pollute the environment. It is astonishing that these devices continue to be manufactured and deployed, but they are sufficiently pervasive that we must design our protocols around their limitations.”

– Network Security: Private Communication in a Public World
Obtaining Human’s Private Key

• Encrypt the private key using the password as a key
  – Put the encrypted private key on a floppy or mag-stripe card
  – Post the encrypted private key in a public place
  – Post the encrypted private key on a server that limits pwd guessing with a careful protocol
• Put the private key on a smart card
Credentials Download

- Have server store $Y = \{\text{priv key}\} \text{pwd}$
- Could send $Y$ in msg 4 in EKE, SPEKE, or PDM. But can do it in 2 msgs!
- Simplification for credential download:
  - No reason to authenticate Bob, or even Alice
- Other features we could do
  - salt
  - saving server computation
4-msg credentials download

share weak secret $W = h(pwd)$

Alice

choose A

“Alice”, $\{g^A \mod p\}W$ → Bob

store

$Y = \{\text{priv}\}pwd$

choose B, challenge C1

choose C2

$\{g^B \mod p\}W$ ←

$K = g^{AB} \mod p$

$h(K)$

→

$\{Y\}K$ ←
2-msg credentials download

Alice  agree on g,p  directory

\[ X = h(\text{pwd}) \]
\[ Y = \{\text{priv key}\}\text{pwd} \]

“Alice”, \( \{g^A \mod p\}X \)

\[ g^B \mod p, \{Y\}g^{AB} \]

slight disadvantage: unaudited on-line guess
Save Bob work: Reuse B

- So store, for Alice (EKE variant)
  - username
  - \(Y = \{\text{priv}\}\text{pwd}\)
  - \(B\)
  - \(\{g^B \mod p\}W\)
  - \(W\)

- Can’t use \(B\) with multiple users with EKE, but can with PDM or SPEKE
Properties of these

- Vulnerabilities to
  - eavesdropper
  - someone impersonating Alice
  - someone impersonating Bob
  - server database theft
So what can we do about server database theft?

- “augmented” EKE (and SPEKE and PDM)
- Augmented EKE came up with the idea, but the protocol is complicated
- We’ll show something simpler that has the same properties (and fewer msgs)
- Main idea: store something at server that can authenticate pwd
- Don’t ever give server a pwd-equiv
Augmented PDM

compute \( p = f(pwd) \), \( W = h(pwd) \)  

Alice  \( \rightarrow \) Bob  
stores \( p, 2^W \mod p \)

choose A  

“Alice”, \( 2^A \mod p \)  \( \rightarrow \) choose B  

\( 2^B \mod p, h(2^{AB} \mod p, 2^{BW} \mod p) \)

\( h'(2^{AB} \mod p, 2^{BW} \mod p) \)
More efficient (for Bob) augmented authentication

- Use RSA key instead of Diffie-Hellman
- Like credentials download, but use RSA key for augmented property (and get credentials download as bonus!)
RSA-Augmented EKE

Alice

compute $W = h(pwd)$

choose A

\[ "Alice", \{g^A \mod p\} \to \text{choose B} \]

\[ g^B \mod p, \{Y\} (g^{AB} \mod p) \]

\[ [h (g^{AB} \mod p)]\text{signed} \]

Bob

stores $W$

$Y = \{\text{priv}\}pwd$

public key
SRP (Secure Remote Pwd)

compute $W = h(pwd)$

Alice

choose $A$

“Alice”, $g^A \mod p$

Bob

stores $g^W \mod p$

$g^B + g^W \mod p$, $u$, $c1$

choose $B$, challenge $c1$, 32-bit # $u$

$K = g^{B(A+uW)} \mod p$

${c1}K$, $c2$

${c2}K$
SRP

• How does each side compute $K$?
• What’s $u$? (without it, someone that has stolen Bob’s database and therefore knows $g^W \mod p$ can impersonate Alice)
  – assume $u$ known in advance, or doesn’t exist
  – instead of sending $g^A \mod p$, send $g^{A-uW} \mod p$
  – Get this by raising $g^W$ to the $u$, getting $g^{uW}$
    Then compute multiplicative inverse of $g^{uW}$ and multiply that by $g^A$