Security Handshake Pitfalls
Login with Shared Secret: Variant 1

B: $R$, A: $K_{AB}\{R\}$, where $K\{}$ can be hash

- authentication not mutual
- connection hijacking
- off-line password attack
- compromise of database at Bob $\rightarrow$ impersonate Alice
Login with Shared Secret: Variant 2

B: $K_{AB}\{R\}$, A: $R$ where $K\{}$ is reversible (DES)

- T: get $K$ without eavesdropping $\implies$ off-line guessing
- weakness of Kerberos 4
- if $R$ has non-random part (e.g., timestamp), Alice can authenticate Bob
Login with Shared Secret: One Way

A: $K_{\text{Alice-Bob}} \{\text{timestamp}\}$

- requires synchronized clocks
- piggyback on password scheme
- stateless
- replay attacks remember messages within clock skew window
- replay attack: several servers with same secret include server name
- need to protect Bob’s clock from being set back secure NTP

use MD instead of encryption include timestamp in the clear
One-Way Public Key

A: hi; B: $R$; A: $[R]_{Alice}$ $\implies$ A signs $R$
A: hi; B: $\{R\}_{Alice}$; A: $R$ $\implies$ A signs $R$

- database at B only write-locked, not read-locked
- either signature (DSS, RSA) or encryption (RSA)
- can trick Alice into signing or decrypting message
- $\implies$ new protocol can compromise old!
- $\implies$ impose structure on message for different uses $\implies$ PKCS

November 9, 2000
Lamport’s Hash

- safe from eavesdropping, database reading
- no public key cryptography
- Alice (human + workstation): password
- Bob (server): username, $n$ (decremented on login), $\text{hash}^n(pw)$

Authentication:

- Alice: name $\rightarrow$ Bob; Bob: $n \rightarrow$ Alice
- Alice: send $x = \text{hash}^{n-1}(pw)$
- Bob: compare $\text{hash}(x)$ with database
- Bob: store new value
- new password: transmit unencrypted
Lamport’s Hash, Salted

- random number $r$ (seed, salt), stored at Bob
- transmit $\text{hash}^n(p|r)$
- different $r$ for different servers
- re-install with different seed value
- avoids precomputation of hashes from dictionary, comparing with database
Lamport’s Hash – Small $n$ Attack

- no mutual authentication
- Bob sends small $n$, say, 50
- Alice sends $\text{hash}^{50}$
- $\Rightarrow$ Bob can generate $\text{hash}^{51}, \text{hash}^{52}, \ldots$
- $\Rightarrow$ Alice has to check if next lower $n$

pencil-and-paper
S/KEY and OTP

• Karn (Bellcore): S/KEY

• RFC 2289 (Feb. 1998)
  – Lamport with alphanumeric salt
  – hash: MD4, MD5, SHA1
  – challenge: \( \text{otp-md5} \ n \ seed \)
  – 64-bit hash: \( \text{MD5(pw} \ | \ \text{seed})^\text{XOR} \rightarrow 64\text{-bits} \)
  – use either 16 hex digits or six words (1 to 4 letters, 11 bits) for key
  – race condition: finish before legitimate user

November 9, 2000
Mutual Authentication: Shared Secret (simplified)

\[ A \rightarrow B \quad \text{I’m Alice, } R_2 \]
\[ B \rightarrow A \quad R_1, K_{AB}\{R_2\} \]
\[ A \rightarrow B \quad K_{AB}\{R_1\} \]
Mutual Authentication – Reflection attack

\[
T \rightarrow B \quad \text{I’m Alice, } R_2
\]
\[
B \rightarrow T \quad R_1, K_{AB}\{R_2\}
\]

Second login by Trudy:

\[
T \rightarrow B \quad \text{I’m Alice, } R_1
\]
\[
B \rightarrow T \quad R_3, K_{AB}\{R_1\}
\]

Fixes:

- different keys for Alice, Bob (derived key) \(\Rightarrow\) T can’t get B to encrypt something using A’s key
- different-type challenges for initiator and responder
- “initiator first to prove identity”
- password guessing: don’t reveal \(K(R), R\) chosen by T
Mutual Authentication: Public Keys

\[
\begin{align*}
A & \rightarrow B \quad \text{I’m Alice, } \{ R_2 \}_B \\
B & \rightarrow A \quad R_2, \{ R_1 \}_A \\
A & \rightarrow B \quad R_1
\end{align*}
\]

variant: sign instead of encrypt

- get *signed* public key (third party, Alice) from Bob
- Bob stores his public key encrypted with Alice’s password
Mutual Authentication: Timestamps (Shared Secret)

\[ A \rightarrow B \quad \text{I’m Alice, } K_{AB}\{t\} \]
\[ B \rightarrow A \quad K_{AB}\{t + 1\} \]

\( t + 1 \) ➤ Trudy can impersonate Alice ➤ include direction flag
Session Keys

- limits exposure of secrets to semi-trusted components
  - shared secrets
  - public keys
  - Bob knows Alice’s public key, Alice knows private key
  - Alice knows password, Bob knows $n$ and $\text{hash}^n(pw)$
Session Key: Shared Secret

\[ A \rightarrow B \quad \text{I’m Alice} \]
\[ B \rightarrow A \quad R \]
\[ A \rightarrow B \quad K_{AB}\{R\} \]

- use \((K_{AB} + 1)\{R\}\) as session key or \(f(K_{AB})\{R\}\)
- \(K_{AB}(R + 1)\) bad \(\Rightarrow\) Trudy can record and then challenge with \(R + 1\)
- \(\Rightarrow\) not quantity encrypted with \(K_{AB}\)
Session Key: Two-Way Public Key

\[ A \rightarrow B: \{R\}_B \]

- weakness: T can send own \{R\} to B

\[ A \rightarrow B: [\{R\}_B]_A \]

- can record conversation, break into B, decrypt
- Alice forgets \( R \) \( \Rightarrow \) overrunning A doesn’t help

A: \( R_1 \), B: \( R_2 \)

\[ A \rightarrow B: \{R_1\}_B; B \rightarrow A: \{R_2\}_A \Rightarrow \text{key } R_1 \oplus R_2 \]

- T needs to overrun both
- T needs to decrypt one \( \Rightarrow \) no need to sign

Diffie-Hellman with signing \( \Rightarrow \) no bucket-brigade attack
Privacy and Integrity

- replay attack ➔ long sequence numbers
- sequence number space rollover ➔ key rollover
Mediated Authentication

- KDC sends shared session key encrypted with destination key
- Avoid race conditions: KDC sends “ticket” to A

\[ K_A\{K_{AB}\} \]

\[ K_B\{K_{AB}\} \]
Needham-Schroeder

- **nonce**: number used once ⇔ seq. no., random number

1. $A \rightarrow KDC: N_1$, Alice wants Bob

2. $K_A \{ N_1, \text{“Bob”, ticket} \} \rightarrow N_1$ to authenticate KDC
ticket = $K_B \{ K_{AB}, \text{“Alice”} \}$ ⇔ KDC ensures Bob that it’s Alice

3. $A \rightarrow B$: challenge Bob with $K_{AB} \{ N_2 \}$, send ticket

4. $B \rightarrow A$: $K_{AB} \{ N_2 - 1, N_3 \}$ ⇔ B proves knowledge of $K_{AB}$

5. $A \rightarrow B$: $K_{AB} \{ N_3 - 1 \}$ ⇔ A proves knowledge of $K_{AB}$
Needham-Schroeder: Reflection Attack

\[ B \rightarrow A: K_{AB}\{N_2 - 1, N_3\} \]

- assume: \( N_i \) multiple of encryption blocksize
- ECB message splicing: put together own plus revealed
- with CBC, no need to decrement \( N_2, N_3 \)
Needham-Schroeder: Limit Compromise

- Trudy steals Alice’s key \(\rightarrow\) can impersonate Alice until key change.
- Alice changes key \(\rightarrow\) ticket to Bob stays valid
- also: T steals old key of Alice
- fix:
  1. \(A \rightarrow B:\) hello!?
  2. \(B \rightarrow A: K_B\{N_B\}, N_B\) made part of ticket \(\rightarrow\) B knows

November 9, 2000
Otway-Rees

- 5 messages, no use of stale tickets
- Suspicious party should generate challenge

1. Nonce $N_C$

2. KDC checks if $N_C$ the same in both $\rightarrow$ Bob $\checkmark$

3. Give ticket; ensures that KDC and Bob are legit

4. B hands (unreadable to B) ticket to A

5. A proves knowledge of $K_{AB}$; A trusts KDC to authenticate B
Kerberos V4

- based on Needham-Schroeder, but with timestamps
- save exchange of nonces
Bellovin-Merritt

- prevent password guessing when T has $R, K\{R\}$
- eavesdropping or address faking of A, B
- Diffie-Hellman exchange, encrypted with shared secret
- agree on common key
- finally, prove possession of common key
- can’t guess key from D-H: random numbers!
- $K$ is just session key
- avoid reflection attack
**Bellovin-Merritt, with Hash**

- Bob only stores hash of A’s password and private key encrypted with password
- $K_{AB} = \text{hash}(pw)$
- D-H ▶️ shared secret $K$ based on hash
- Alice proves knowledge of $K$ (=hash) by encrypting $R$
- Bob encrypts Alice’s encrypted private key
- Alice signs $R$, Bob verifies using public key
- Bob needs to keep encrypted password secret!
Avoiding Password Guessing

- Don’t send encrypted version and plaintext
- protection against active and passive attacks
- another attack: impersonate Bob

1. send to anyone  ➔ active attack
2. prove knowledge of Alice’s secret
3. encrypt (2) via session key
4. encrypt (2) with secret or public key for Bob
5. use Bellovin-Merritt, then (1) or (2)
Nonce Types

- timestamp ➔ synchronized clocks
- large random number ➔ cannot predict, guess
- sequence number ➔ non-volatile state
Nonce Types: Sequence Numbers

\[
\begin{align*}
A & \rightarrow B \quad \text{I’m Alice} \\
B & \rightarrow A \quad K_{AB}\{R\} \\
B & \rightarrow A \quad (K_{AB} + 1)\{R\}
\end{align*}
\]

\(R\) just has to be non-repeating
Random Numbers

needed for:

- cryptographic keys
- challenges
- IVs
- per-message secrets for El-Gamal/DSS

random: unpredictable (\(\pi\)) or unguessable

defaultrandom: deterministic algorithm

- thermal (noise diode), video, audio noise
- keyboard timing, disk seek times
- current clock bits
• process number, system load, number of users, …
• packets seen, sent
• hardware id
Generating Random Numbers

- start with random seed, then hash
- pseudorandom number generator:
  1. hash of seed
  2. hash of (previous output | seed)
Performance

**Computation:** bytes hashed, private key > public key; parallelization?

**Delay:** message exchanges

**Cacheability:** for repeated authentication