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 ➠ impersonate Alice

Login with Shared Secret: Variant 2

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

- T: get K without eavesdropping ➠ 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: $K_{\text{Alice}-\text{Bob}}$ {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: $[R]_{\mathrm{Alice}}$ ➠ A signs R
A: hi; B: $\{R\}_{\mathrm{Alice}}$; A: R ➠ 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
- ➠ new protocol can compromise old!
- impose structure on message for different uses ➠ PKCS

Lamport's Hash

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

Authentication:

- Alice: name $\rightarrow$ Bob; Bob: $n \rightarrow$ Alice
- Alice: send x = hash^{n-1}(pw)
- Bob: compare 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 $\mathrm{hash}^n(p|r)$
- 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 hash^{50}
- ➠ Bob can generate $\mathrm{hash}^{51}, \mathrm{hash}^{52}, \ldots$
- ➠ 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: `otp-md5` n *seed*
  - 64-bit hash: MD5(pw | seed) $\stackrel{XOR}{\rightarrow}$ 64-bits
  - use either 16 hex digits or six words (1 to 4 letters, 11 bits) for key
  - race condition: finish before legitimate user

Mutual Authentication: Shared Secret (simplified)

$A \rightarrow B$	I'm Alice, R_2
$B \rightarrow A$	$R_1, K_{\mathrm{AB}}\{R_2\}$
$A \rightarrow B$	$K_{\mathrm{AB}}\{R_1\}$

Mutual Authentication – Reflection attack

$T \rightarrow B$ I'm Alice, R_2

$B \rightarrow T$ $R_1, K_{\mathrm{AB}}\{R_2\}$

Second login by Trudy:

$T \rightarrow B$ I'm Alice, R_1

$B \rightarrow T$ $R_3, K_{\mathrm{AB}}\{R_1\}$

Fixes:

- different keys for Alice, Bob (derived key) ➠ 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

$A \rightarrow B$ I'm Alice, $\{R_2\}_{\mathrm{B}}$

$B \rightarrow A$ R_2, $\{R_1\}_{\mathrm{A}}$

$A \rightarrow B$ R_1

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)

$A \rightarrow B$	I'm Alice, $K_{AB}\{t\}$
$B \rightarrow A$	$K_{AB}\{t+1\}$

$t+1$ ➠ 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 hash^n(pw)

Session Key: Shared Secret

$$
\begin{array}{ll}
A \to B & \text{I'm Alice} \\
B \to A & R \\
A \to B & K_{AB}\{R\}
\end{array}
$$

- use $(K_{AB} + 1)\{R\}$ as session key or $f(K_{AB})\{R\}$
- $K_{AB}(R + 1)$ bad ➠ Trudy can record and then challenge with $R + 1$
- ➠ not quantity encrypted with K_{AB}

Session Key: Two-Way Public Key

$A \to B$: $\{R\}_B$

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

$A \to B$: $[\{R\}_B]_A$

- can record conversation, break into B, decrypt
- Alice forgets R ➠ overrunning A doesn't help

A: R_1, B: R_2
$A \to B$: $\{R_1\}_B$; $B \to A$: $\{R_2\}_A$ ➠ key $R1 \oplus R2$

- T needs to overrun both
- T needs to decrypt one ➠ no need to sign

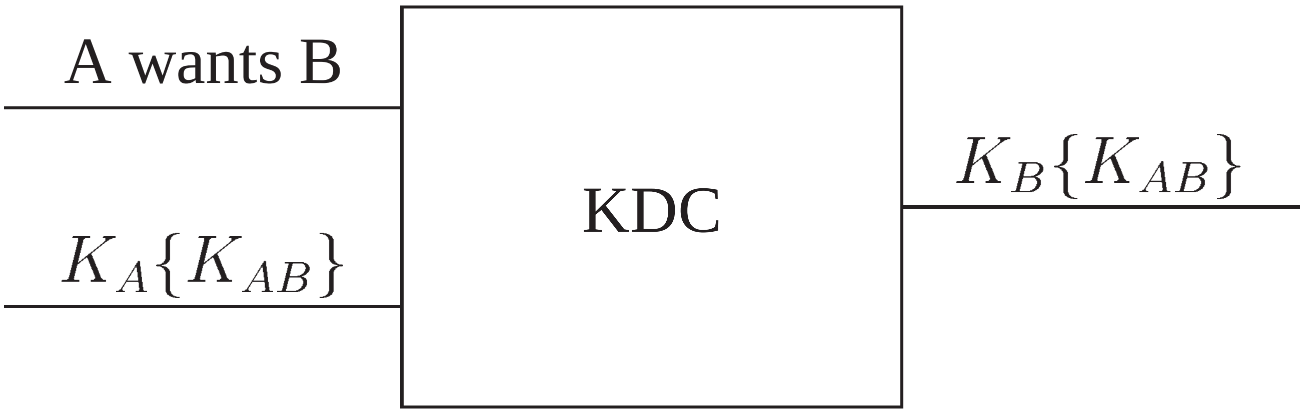
Diffie-Hellman with signing ➠ no bucket-brigade attack

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



Needham-Schroeder

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

1. $A \rightarrow$ KDC: N_1, Alice wants Bob
2. $K_A\{N_1, \text{“Bob”}, \text{ticket}\}$ ➠ 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

$B \to A$: $K_{AB}\{N_2 - 1, N_3\}$

- 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 ➠ can impersonate Alice until key change.
- Alice changes key ➠ ticket to Bob stays valid
- also: T steals old key of Alice
- fix:

  1. $A \to B$: hello!?
  2. $B \to A$: $K_B\{N_B\}$, N_B made part of ticket ➠ B knows

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 ➠ Bob $\surd$
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} = 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

$$
\begin{array}{ll}
A \rightarrow B & \text{I'm Alice} \\
B \rightarrow A & K_{AB}\{R\} \\
B \rightarrow A & (K_{AB}+1)\{R\}
\end{array}
$$

R just has to be non-repeating

Random Numbers

needed for:

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

random: unpredictable (π) or unguessable

pseudorandom: 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