Notation

- $X \rightarrow Y : \{ Z \mid|\ W \} k_{X,Y}$
  - $X$ sends $Y$ the message produced by concatenating $Z$ and $W$ enciphered by key $k_{X,Y}$, which is shared by users $X$ and $Y$

- $A \rightarrow T : \{ Z \} k_A \mid| \{ W \} k_{A,T}$
  - $A$ sends $T$ a message consisting of the concatenation of $Z$ enciphered using $k_A$, $A$’s key, and $W$ enciphered using $k_{A,T}$, the key shared by $A$ and $T$

- $r_1$, $r_2$ nonces (nonrepeating random numbers)
Symmetric Key Exchange

• Bootstrap problem: how do Alice, Bob begin?
  • Alice can’t send it to Bob in the clear!
• Assume trusted third party, Cathy
  • Alice and Cathy share secret key $k_A$
  • Bob and Cathy share secret key $k_B$
• Use this to exchange shared key $k_s$
Simple Protocol

Alice \rightarrow Cathy

\{ \text{request for session key to Bob} \} k_A

Alice \leftarrow Cathy

\{k_s\} k_A \ || \ \{k_s\} k_B

Alice \rightarrow Bob

\{k_s\} k_B
Problems

• How does Bob know he is talking to Alice?
  • Replay attack: Eve records message from Alice to Bob, later replays it; Bob may think he’s talking to Alice, but he isn’t
  • Session key reuse: Eve replays message from Alice to Bob, so Bob re-uses session key

• Protocols must provide authentication and defense against replay
Session, Interchange Keys

- Alice wants to send a message $m$ to Bob
  - Assume public key encryption
  - Alice generates a random cryptographic key $k_s$ and uses it to encipher $m$
    - To be used for this message only
    - Called a session key
  - She enciphers $k_s$ with Bob’s public key $k_B$
    - $k_B$ enciphers all session keys Alice uses to communicate with Bob
    - Called an interchange key
  - Alice sends \{ $m$ \} $k_s$ \{ $k_s$ \} $k_B$
Benefits

• Limits amount of traffic enciphered with single key
  • Standard practice, to decrease the amount of traffic an attacker can obtain

• Prevents some attacks
  • Example: Alice will send Bob message that is either “BUY” or “SELL”. Eve computes possible ciphertexts \{ “BUY” \} \text{k_B} and \{ “SELL” \} \text{k_B}. Eve intercepts enciphered message, compares, and gets plaintext at once
Kerberos

• Authentication system
  • Based on Needham-Schroeder with Denning-Sacco modification
  • Central server plays role of trusted third party ("Cathy")

• Ticket
  • Issuer vouches for identity of requester of service

• Authenticator
  • Identifies sender
Idea

• User $u$ authenticates to Kerberos server
  • Obtains ticket $T_{u,TGS}$ for ticket granting service (TGS)

• User $u$ wants to use service $s$:
  • User sends authenticator $A_{u}$, ticket $T_{u,TGS}$ to TGS asking for ticket for service
  • TGS sends ticket $T_{u,s}$ to user
  • User sends $A_{u}$, $T_{u,s}$ to server as request to use $s$

• Details follow
Ticket

- Credential saying issuer has identified ticket requester
- Example ticket issued to user $u$ for service $s$

$$T_{u,s} = s || \{ u || u's \text{ address} || \text{valid time} || k_{u,s} \} k_s$$

where:
- $k_{u,s}$ is session key for user and service
- Valid time is interval for which ticket valid
- $u's$ address may be IP address or something else
  - Note: more fields, but not relevant here
Authenticator

- Credential containing identity of sender of ticket
  - Used to confirm sender is entity to which ticket was issued
- Example: authenticator user $u$ generates for service $s$

$$A_{u,s} = \{ u \mid\mid \text{generation time} \mid\mid k_t \} \ k_{u,s}$$

where:
- $k_t$ is alternate session key
- Generation time is when authenticator generated
  - Note: more fields, not relevant here
Protocol

\[
\begin{align*}
\text{user} & \quad \text{user} \oplus TGS \quad \rightarrow \quad \text{AS} \\
\text{AS} & \quad \left\{ k_{u,TGS} \right\} k_u \oplus T_u,TGS \quad \rightarrow \quad \text{user} \\
\text{user} & \quad \text{service} \oplus A_{u,TGS} \oplus T_u,TGS \quad \rightarrow \quad \text{TGS} \\
\text{user} & \quad \text{user} \oplus \left\{ k_{u,s} \right\} k_{u,TGS} \oplus T_u,s \quad \rightarrow \quad \text{TGS} \\
\text{user} & \quad A_{u,s} \oplus T_u,s \quad \rightarrow \quad \text{service} \\
\text{user} & \quad \left\{ t + 1 \right\} k_{u,s} \quad \rightarrow \quad \text{service}
\end{align*}
\]
Analysis

- First two steps get user ticket to use TGS
  - User $u$ can obtain session key only if $u$ knows key shared with AS
- Next four steps show how $u$ gets and uses ticket for service $s$
  - Service $s$ validates request by checking sender (using $A_{u,s}$) is same as entity ticket issued to
  - Step 6 optional; used when $u$ requests confirmation
Problems

• Relies on synchronized clocks
  • If not synchronized and old tickets, authenticators not cached, replay is possible

• Tickets have some fixed fields
  • Dictionary attacks possible
  • Kerberos 4 session keys weak (had much less than 56 bits of randomness); researchers at Purdue found them from tickets in minutes
Public Key Key Exchange

• Here interchange keys known
  • $e_A, e_B$ Alice and Bob’s public keys known to all
  • $d_A, d_B$ Alice and Bob’s private keys known only to owner

• Simple protocol
  • $k_s$ is desired session key

\[ \text{Alice} \xrightarrow{\{ k_s \} e_B} \text{Bob} \]
Problem and Solution

• Vulnerable to forgery or replay
  • Because $e_B$ known to anyone, Bob has no assurance that Alice sent message

• Simple fix uses Alice’s private key
  • $k_s$ is desired session key

\[
\text{Alice} \quad \{ \{ k_s \} d_A \} e_B \quad \rightarrow \quad \text{Bob}
\]
Notes

- Can include message enciphered with $k_s$
- Assumes Bob has Alice’s public key, and *vice versa*
  - If not, each must get it from public server
  - If keys not bound to identity of owner, attacker Eve can launch a *man-in-the-middle* attack (next slide; Cathy is public server providing public keys)
    - Solution to this (binding identity to keys) discussed later as public key infrastructure (PKI)
Man-in-the-Middle Attack

Alice → send Bob’s public key → Eve

Eve → send Bob’s public key → Cathy

Eve → intercepts request → Cathy

Eve → intercepts message → Bob

Alice → $e_E$ → Eve

Alice → $\{k_s\}e_E$ → Bob

Eve → $\{k_s\}e_B$ → Bob
Diffie-Hellman

• Compute a common, shared key
  • Called a *symmetric key exchange protocol*

• Based on discrete logarithm problem
  • Given integers $n$, $g$ and prime number $p$, compute $k$ such that $n = g^k \mod p$
  • Solutions known for small $p$
  • Solutions computationally infeasible as $p$ grows large
Algorithm

• Constants: prime $p$, integer $g \neq 0, 1, p-1$
  • Known to all participants
• Alice chooses private key $k_{\text{Alice}}$, computes public key $K_{\text{Alice}} = g^{k_{\text{Alice}}} \mod p$
• Bob chooses private key $k_{\text{Bob}}$, computes public key $K_{\text{Bob}} = g^{k_{\text{Bob}}} \mod p$
• To communicate with Bob, Anne computes $K_{\text{Alice,Bob}} = K_{\text{Bob}}^{k_{\text{Alice}}} \mod p$
• To communicate with Anne, Bob computes $K_{\text{Bob,Alice}} = K_{\text{Alice}}^{k_{\text{Bob}}} \mod p$
• It can be shown $K_{\text{Alice,Bob}} = K_{\text{Bob,Alice}}$
Example

- Assume $p = 121001$ and $g = 6981$
- Alice chooses $k_{Alice} = 526784$
  - Then $K_{Alice} = 6981^{26874} \mod 121001 = 22258$
- Bob chooses $k_{Bob} = 5596$
  - Then $K_{Bob} = 6981^{5596} \mod 121001 = 112706$
- Shared key:
  - $K_{Bob}^{k_{Alice}} \mod p = 112706^{26874} \mod 121001 = 78618$
  - $K_{Alice}^{k_{Bob}} \mod p = 22258^{5596} \mod 121001 = 78618$
Problems

- Using cipher requires knowledge of environment, and threats in the environment, in which cipher will be used
  - Is the set of possible messages small?
  - Can an active wiretapper rearrange or change parts of the message?
  - Do the messages exhibit regularities that remain after encipherment?
  - Can the components of the message be misinterpreted?
Attack #1: Precomputation

- Set of possible messages $M$ small
- Public key cipher $f$ used
- Idea: precompute set of possible ciphertexts $f(M)$, build table $(m, f(m))$
- When ciphertext $f(m)$ appears, use table to find $m$
- Also called forward searches
Example

• Cathy knows Alice will send Bob one of two messages: enciphered BUY, or enciphered SELL
• Using public key $e_{Bob}$, Cathy precomputes
  $$m_1 = \{ \text{BUY} \} e_{Bob}, \quad m_2 = \{ \text{SELL} \} e_{Bob}$$
• Cathy sees Alice send Bob $m_2$
• Cathy knows Alice sent SELL
May Not Be Obvious

• Digitized sound
  • Seems like far too many possible plaintexts, as initial calculations suggest $2^{32}$ such plaintexts
  • Analysis of redundancy in human speech reduced this to about 100,000 ($\approx 2^{17}$), small enough for precomputation attacks
Misordered Blocks

- Alice sends Bob message
  - $n_{Bob} = 262631$, $e_{Bob} = 45539$, $d_{Bob} = 235457$
- Message is TOMNOTANN (191412 131419 001313)
- Enciphered message is 193459 029062 081227
- Eve intercepts it, rearranges blocks
  - Now enciphered message is 081227 029062 193459
- Bob gets enciphered message, deciphers it
  - He sees ANNOTTOM, opposite of what Alice sent
Statistical Regularities

• If plaintext repeats, ciphertext may too
• Example using AES-128:
  • Input image: Hello world!
  • corresponding output image:
    
• Note you can still make out the words
• Fix: cascade blocks together (chaining); more details later
Type Flaw Attacks

• Assume components of messages in protocol have particular meaning

• Example: Otway-Rees:

  Alice \[ n \| Alice \| Bob \| \{ r_1 \| n \| Alice \| Bob \} k_A \]

  Cathy \[ n \| Alice \| Bob \| \{ r_1 \| n \| Alice \| Bob \} k_A \| \{ r_2 \| n \| Alice \| Bob \} k_B \]

  Cathy \[ n \| \{ r_1 \| k_s \} k_A \| \{ r_2 \| k_s \} k_B \]

  Alice \[ n \| \{ r_1 \| k_s \} k_A \]

  Cathy \[ n \| \{ r_1 \| k_s \} k_B \]

  Bob
The Attack

• Ichabod intercepts message from Bob to Cathy in step 2
• Ichabod *replays* this message, sending it to Bob
  • Slight modification: he deletes the cleartext names
• Bob *expects* \( n \| \{ r_1 \| k_s \} k_A \| \{ r_2 \| k_s \} k_B \)
• Bob *gets* \( n \| \{ r_1 \| n \| Alice \| Bob \} k_A \| \{ r_2 \| n \| Alice \| Bob \} k_B \)
• So Bob sees \( n \| Alice \| Bob \) as the session key — and Ichabod knows this
• When Alice gets her part, she makes the same assumption
• Now Ichabod can read their encrypted traffic
Solution

- Tag components of cryptographic messages with information about what the component is
  - But the tags themselves may be confused with data ...
What These Mean

• Use of strong cryptosystems, well-chosen (or random) keys not enough to be secure

• Other factors:
  • Protocols directing use of cryptosystems
  • Ancillary information added by protocols
  • Implementation (not discussed here)
  • Maintenance and operation (not discussed here)
Networks and Cryptography

- ISO/OSI model
- Conceptually, each host communicates with peer at each layer
Link and End-to-End Protocols

Link Protocol

End-to-End (or E2E) Protocol