Chapter 2

Basic Cryptography & Key Exchange Protocols

Nguyen Ngoc Thien An
Department of Information Systems
Faculty of Computer Science & Engineering
HCMC University of Technology
annguyen@cse.hcmut.edu.vn
Outline

- Cryptography Overview
- Classic Cryptography
- Modern Cryptography
- Key Channel
- Perfect Encryption
- Dolev-Yao Threat Model
- Key Exchange Protocols
Outline

- Cryptography Overview
  - Terminology
  - Classification
  - Cryptosystems
  - Kerckhoffs Principle
  - Brute-force Attack
  - Definition of Secure
Terminology (1)

- **Cryptology** is the art and science of making and breaking “secret codes”.
- **Cryptography** is the study of the making of “secret codes”.
- **Cryptanalysis** is the study of how to crack encryption algorithms or their implementations.
- **Cryptography** and **cryptology** are usually used interchangeably to refer to the combined study of cryptography and cryptanalysis.
Terminology (2)

- **Encryption**: the process of the making of “secret codes”.
- **Decryption**: the process of the breaking of “secret codes”.
- **Cipher**: an algorithm for performing encryption or decryption.
- **Plaintext**: a message input to an encryption algorithm and needing protecting. It may or may not be *intelligible*.
  - **Cleartext** is an intelligible plaintext.
- **Ciphertext**: the result of encryption performed on plaintext using an algorithm. Ciphertext is not *intelligible*.
- **Cryptosystem** = encryption + decryption algorithms.
  - Encryption and decryption processes need **keys**.
Classification

- **Symmetric cryptosystems** use the same key for (en/decrypt) algorithms.
  - Symmetric key = Shared-secret-key.

- **Asymmetric cryptosystems** (Public-key cryptosystems) use two different keys for (en/decrypt) algorithms.
  - Public key & Private key.
Cryptosystems

- **Symmetric cryptosystems**: $K_E = K_D$
- **Asymmetric cryptosystems**: $K_E \neq K_D$
Kerckhoffs Principle

- **Kerckhoffs principle** for cryptography: “A cipher must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience”.

- The cryptographic community will not accept an algorithm as secure until it has withstood extensive analyses by many cryptographers over an extended period of time.
  - The belief is that “more eyeballs” are more likely to expose security flaws.

- **Kerckhoffs principle** for security: “The security design itself is open”.

8/10/2014
Brute-force Attack

- A **brute-force attack** (or **exhaustive key search**): systematically checking all possible keys or passwords until the correct one is found.
  - In theory, it can be used against any encrypted data.
  - In the worst case, this would involve traversing the entire search space.

- **1998**, the Electronic Frontier Foundation (EFF) built:
  - A special-purpose key cracking machine for attacking DES.
  - 43,200 processors, each of which ran at 40 MHz and was capable of testing about 2.5 million keys per second.

- Extrapolate to a state-of-the-art PC with a single 4 GHz processor:
  - Could test fewer than $2^{30}$ keys/second.
Definition of Secure

- **A cryptosystem is secure** if the best-known attack requires as much work as an exhaustive key search, that is, there is no short-cut attack.
  - A secure cryptosystem with a small number of keys vs. an insecure cryptosystem with a large number of keys.

- In practice, we must select a cipher that:
  - Is secure (in the sense of our definition)
  - Has a large enough key space so that an exhaustive key search is impractical.
    → Both factors are necessary.
Outline

- Cryptography Overview
- Classic Cryptography
- Modern Cryptography
- Key Channel
- Perfect Encryption
- Dolev-Yao Threat Model
- Key Exchange Protocols
Outline

- Classic Cryptography
  - Simple Substitution Cipher
  - Double Transposition Cipher
  - One-time Pad
  - Codebook Cipher
Simple Substitution Cipher

- The message is encrypted by substituting the letter of the alphabet \( n \) places ahead of the current letter.

- E.g.: \( n = 3 \) (Caesar’s cipher):

  a b c d e f g h i j k l m n o p q r s t u v w x y z
  D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

- The possible keys are \( n \in \{0, 1, 2, ..., 25\} \rightarrow \) the worst: 26 tries; the average: 13 tries.
Double Transposition Cipher

- Write the plaintext into an array of a given size.
- **Permute the rows and columns** according to specified permutations.

E.g.:
- 3×4 array
- Rows: (1, 2, 3) → (3, 2, 1)
- Columns: (1, 2, 3, 4) → (4, 2, 1, 3)

$$\begin{bmatrix}
\text{a t t a} \\
\text{c k a t} \\
\text{d a w n}
\end{bmatrix} \rightarrow 
\begin{bmatrix}
\text{d a w n} \\
\text{c k a t} \\
\text{a t t a}
\end{bmatrix} \rightarrow 
\begin{bmatrix}
\text{n a d w} \\
\text{t k c a} \\
\text{a t t a}
\end{bmatrix}$$
One-time Pad (1)

- Also the Vernam cipher.

Encryption

- Convert the letters to the bit string → the plaintext.
- Create a key consisting of a randomly selected string of bits that is the same length as the message.
- The key is XORed with the plaintext → the ciphertext.
- Convert the ciphertext bits back into letters.

<table>
<thead>
<tr>
<th>plaintext:</th>
<th>h e i l h i t l e r</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>001 000 010 100 001 010 111 100 000 101</td>
</tr>
</tbody>
</table>

| key:       | 111 101 110 101 111 100 000 101 110 000 |

<table>
<thead>
<tr>
<th>ciphertext:</th>
<th>s r l h s s t h s s r</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>110 101 100 001 110 110 111 001 110 101</td>
</tr>
</tbody>
</table>
One-time Pad (2)

- **Decryption**
  - Convert the letters to the bit string → the ciphertext.
  - The ciphertext is XORed with the same key → the plaintext (x ⊕ y ⊕ y = x).
  - Convert the plaintext bits back into letters.

<table>
<thead>
<tr>
<th>ciphertext:</th>
<th>s</th>
<th>r</th>
<th>l</th>
<th>h</th>
<th>s</th>
<th>s</th>
<th>t</th>
<th>h</th>
<th>s</th>
<th>r</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>110</td>
<td>101</td>
<td>100</td>
<td>001</td>
<td>110</td>
<td>110</td>
<td>111</td>
<td>001</td>
<td>110</td>
<td>101</td>
</tr>
<tr>
<td>key:</td>
<td>111</td>
<td>101</td>
<td>110</td>
<td>101</td>
<td>111</td>
<td>100</td>
<td>000</td>
<td>101</td>
<td>110</td>
<td>000</td>
</tr>
<tr>
<td>plaintext:</td>
<td>001</td>
<td>000</td>
<td>010</td>
<td>100</td>
<td>001</td>
<td>010</td>
<td>111</td>
<td>100</td>
<td>000</td>
<td>101</td>
</tr>
<tr>
<td></td>
<td>h</td>
<td>e</td>
<td>i</td>
<td>l</td>
<td>h</td>
<td>i</td>
<td>t</td>
<td>l</td>
<td>e</td>
<td>r</td>
</tr>
</tbody>
</table>
A classic codebook cipher is, literally, a dictionary-like book containing words and their corresponding codewords.

- E.g.: Encrypt the entire German word Februar by replacing it with the 5-digit “codeword” 13605.

The codebook in Table 2.3 was used for encryption, while a corresponding codebook, arranged with the 5-digit codewords in numerical order, was used for decryption.
Codebook Cipher (2)

- In a codebook, substitutions are for entire words, or even phrases → It is a complex substitution cipher.
- Modern block ciphers use complex algorithms to generate ciphertext from plaintext (and vice versa) but at a higher level, a block cipher can be viewed as a codebook, where each key determines a distinct codebook.
Outline

- Cryptography Overview
- Classic Cryptography
- Modern Cryptography
- Key Channel
- Perfect Encryption
- Dolev-Yao Threat Model
- Key Exchange Protocols
Outline

- Modern Cryptography
  - Symmetric cryptosystems:
    - Stream ciphers: A5/1, RC4
    - Block ciphers: DES, Triple DES, AES
  - Asymmetric cryptosystems:
    - RSA
    - Secret Key Establishment
    - Digital Signatures
    - Digital Certificate
Symmetric Cryptosystems - Stream Ciphers (1)

- Take a key $K$ of $n$ bits in length and stretches it into a long keystream $S$: $\text{StreamCipher}(K) = S$.
- $P = p_0 \ p_1 \ p_2 \ldots p_n$; $S = s_0 \ s_1 \ s_2 \ldots s_n$; $C = c_0 \ c_1 \ c_2 \ldots c_n$
- Encryption: use the keystream to encrypt as the use of the key in a one-time pad cipher.
  \[ c_0 = p_0 \oplus s_0, \ c_1 = p_1 \oplus s_1, \ c_2 = p_2 \oplus s_2, \ldots \]
- Decryption: the same keystream is generated and XORed with the ciphertext.
  \[ p_0 = c_0 \oplus s_0, \ p_1 = c_1 \oplus s_1, \ p_2 = c_2 \oplus s_2, \ldots \]
Symmetric Cryptosystems - Stream Ciphers (2)

- **A5/1**
  - Employ 3 linear feedback *shift registers* (LFSRs).
  - Be representative of a large class of stream ciphers that are based on shift registers and implemented in hardware.
  - Can generate bits at a rate proportional to the clock speed.
  - Be used by GSM cell phones for confidentiality.

- **RC4**
  - Be efficiently implemented in software \( \rightarrow \) RC4 is almost unique among stream ciphers.
  - Be used in many places, including in the Secure Socket Layer, or SSL protocol.
  - Be fairly old and not optimized for 32-bit processors (it’s optimized for ancient 8-bit processors).
Symmetric Cryptosystems - Block Ciphers

- An iterated block cipher splits the plaintext into fixed sized blocks and generates fixed sized blocks of ciphertext.
- The ciphertext is obtained from the plaintext by iterating a function $F$ over some number of rounds.
- The function $F$, which depends on the output of the previous round and the key $K$, is known as a *round function*.
- The design goals for block ciphers are security and efficiency.
- Most popular algorithms: **DES, Triple DES, AES**.
Block Ciphers - DES

- **DES**: Data Encryption Standard.
- Be published in 1977 by the USA’s National Bureau of Standards (NBS) – now known as National Institute of Standards and Technology (NIST).
- Be used as the encryption standard for unclassified data (information not concerned with national security).
- Messages are divided into 64-bit blocks.
- **Key length**: 56 bits → Today, it’s too small → Brute-force attack (some hours).
Block Ciphers - Triple DES

- Improve the key length → Run the DES algorithm a multiple number of times using different keys.
- 1999, **Triple DES** was encouraged to replace **DES**.
- Why not “Double DES”???

**Triple DES:**
- $m$: plaintext; $c$: ciphertext
- $\mathcal{E}_{k_1}$: encryption by key $k_1$; $\mathcal{D}_{k_1}$: decryption by key $k_1$;
- Encryption: $c \leftarrow \mathcal{E}_{k_1}(\mathcal{D}_{k_2}(\mathcal{E}_{k_1}(m)))$
- Decryption: $m \leftarrow \mathcal{D}_{k_1}(\mathcal{E}_{k_2}(\mathcal{D}_{k_1}(c)))$

- Be compatible with DES when $k_1 = k_2$;
- The triple DES can also use three different keys.
Block Ciphers - AES (1)

- **AES: Advanced Encryption Standard** (also known as Rijndael)
  - Jan 1997: NIST announced the initiation of a new symmetric-key block cipher algorithm, AES, as the new encryption standard to replace the DES.
  - Oct 2000: Rijndael designed by 2 Belgium cryptographers (Daemen and Rijmen) was selected.
- Be an iterated block cipher with a variable block size and variable key size.
- The key size and the block size can be independently specified to 128, 192 or 256 bits.
Block Ciphers - AES (2)

- Be secure and can be implemented efficiently.
- The enlarged and variable key and data-block sizes can accommodate a wide spectrum of security strengths for various application needs.
- Replace for multiple encryption (as Triple DES) reduce the cost of managing keys simplify the design of security protocols and systems.
Asymmetric Cryptosystems - Overview

- In Asymmetric cryptosystems (Public-key cryptosystems): public key for encryption, private key for decryption.
Asymmetric Cryptosystems - Overview

- They are based on very special mathematical transformations termed **trap door one-way functions**.
  - A *trap door one-way function* is easy to compute in one direction and hard to compute in other direction
  - E.g.: Generate 2 prime numbers $p$ and $q$, compute $N = pq$: Easy
    With given $N$, find its factors $p$ and $q$: Difficult
Assymmetric Cryptosystems - RSA

- **RSA** is the best known public-key cryptosystem, named after its inventors - Rivest, Shamir and Adleman.

**Basic steps:**

- Key generation:
  - Choose 2 large prime numbers randomly and independently: $p \& q, p \neq q$.
  - Compute: $n = pq$; Euler's phi function: $\phi(n) = (p - 1)(q - 1)$.
  - Choose **public (or encryption) exponent** $e$: $e \in N, 1 < e < \phi(n)$, $e$ and $\phi(n)$ are coprime.
  - Find **private (or decryption) exponent** $d$: $ed \equiv 1 \pmod{\phi(n)}$.
  - The public key (for encryption): $(n, e)$
  - The private key (for decryption): $(n, d)$

- Encryption: $c = m^e \mod n$.
- Decryption: $m = c^d \mod n$.
Assymetric Cryptosystems - Secret key establishment

- Symmetric cryptosystems: faster, more efficient.
- Asymmetric cryptosystems: more secure, no shared key in advance.
- Application: establish secret keys over public communication channels.
  - Hybrid scheme:
    - Asymmetric technique: establish a symmetric key.
    - Symmetric technique: encrypt data by that symmetric key.
Asymmetric Cryptosystem - Digital signatures

- **Digital signatures**: a message signed with a user's private key and can be verified by anyone who has access to the user's public key, thereby proving that the user signed it and that the message has not been tampered with.

- Thus:
  - **Authentication**: Anyone with the user’s public key can verify his digital signature → Easy to verify who is the owner of the message.
  - **Data integrity**: The message cannot be modified without being detected during decryption time → Verify whether the message is modified or not.
  - **Non-repudiation**: Only the holder of the private key can digitally sign → it prevents the sender from claiming that he did not actually send the information.
Assymmetric Cryptosystem - Digital signatures

Original text: Memo: Confidential
Re: Fiscal Review
This quarter's earnings have just come in and...

Signing:

Signed text: Memo: Confidential
Re: Fiscal Review
This quarter's earnings have just come in and...

Verifying:

Verified text: Memo: Confidential
Re: Fiscal Review
This quarter's earnings have just come in and...

Simple digital signatures
In an asymmetric cryptosystem, digital signatures are used for secure communication. The process involves:

1. plaintext
2. hash function applied to plaintext
3. message digest
4. digest signed with private key
5. plaintext + signature

The private key is used for signing, ensuring the security of the digital signature.
Digital Certificate (1)

- A **public key infrastructure**, or **PKI**, is the sum total of everything required to securely use public key crypto.
- A **digital certificate** (or **identity certificate**, or **public key certificate**) is an electronic document which uses a **digital signature** to bind a public key with an identity.
- The certificate can be used to verify that a public key belongs to an individual.
- In most situations the certificate must be signed by a **Certificate Authority**, or **CA**, which acts as a **trusted third party**.
A digital certificate can be revoked if:

- Its private key is compromised.
- The relationship of the public key and the owner is changed.
- A certificate was issued in error.

⇒ Use the **Certificate Revocation List**.

The largest commercial source for certificates is **VeriSign**.
Each digital certificate contains basic information:

- Name and URL of the CA.
- The user’s public key.
- The user’s identity: information such as the name of a person or an organization, their address or their server address, and so forth.
- Expired date.
Outline

- Cryptography Overview
- Classic Cryptography
- Modern Cryptography
- Key Channel
- Perfect Encryption
- Dolev-Yao Threat Model
- Key Exchange Protocols
Key Channel (1)

- Alice and Bob want to make conversation in a secure manner by using symmetric encryption.
- Alice and Bob have not agreed about a secret key.
- A communication channel over which a key is correctly established is called a **key channel**.
- A key channel is a separate channel from a **message channel**.
Key Channel (2)

- Conventional techniques:
  - Relying on an online authentication service (through a Trusted Third Party).
  - This disadvantage limits the scalability of the technique for any open systems applications.

- Public-key techniques (with the hybrid scheme)
  - Asymmetric technique: for the key encryption.
  - Symmetric technique: for data encryption.

- The Quantum Key Distribution Technique: [4] 4.4.5
Key Channel (3)

- Security properties for authenticated key establishment:
  - Only Alice and Bob (or perhaps a principal who is trusted by them) should know K.
  - Alice and Bob should know that the other principal knows K.
  - Alice and Bob should know that K is newly generated.
Outline

- Cryptography Overview
- Classic Cryptography
- Modern Cryptography
- Key Channel
- Perfect Encryption
- Dolev-Yao Threat Model
- Key Exchange Protocols
Perfect Encryption (1)

- **Notation:**
  - $M$: Plaintext
  - $A$: Encryption algorithm
  - $K$: Encryption key
  - $M'$: Ciphertext
  - $A'$: Decryption algorithm
  - $K'$: Decryption key

- **The relationship of $M$ and $M'$:**
  \[
  M' = A(K, M) = \{M\}_K \\
  M = A'(K', M') = A'(K', A(K, M))
  \]

- **Property 1:**
  Without the key $K$ (in the case of a symmetric cryptosystem), or the matching private key of $K$ (in the case of an asymmetric cryptosystem), the ciphertext $\{M\}_K$ does not provide any cryptanalytic means for finding the plaintext message $M$. 
Perfect Encryption (2)

- **Property 2:**
The ciphertext $\{M\}_K$ and maybe together with some known information about the plaintext $M$ do not provide any cryptanalytic means for finding the key $K$ (in the case of a symmetric cryptosystem), or the matching private key of $K$ (in the case of an asymmetric cryptosystem).

- **Property 3:** (for message authentication service)
Without the key $K$, even with the knowledge of the plaintext $M$, it should be impossible for someone to alter $\{M\}_K$ without being detected by the recipient during the time of decryption.
Outline

- Cryptography Overview
- Classic Cryptography
- Modern Cryptography
- Key Channel
- Perfect Encryption
- Dolev-Yao Threat Model
- Key Exchange Protocols
Dolev-Yao Threat Model (1)

- **Dolev and Yao** proposed a **threat model** which has been widely accepted as the standard threat model for cryptographic protocols.

- **Characters of the model:**
  - Alice & Bob: normal users (or principal, entity, agent) who wish to communicate with each other in a secure manner.
  - Trent: a trusted third party.
  - Malice: a bad guy (or attacker, adversary, enemy, intruder, eavesdropper, impostor, etc.).
Dolev-Yao Threat Model (2)
Dolev-Yao Threat Model (3)

- Malice can:
  - He can obtain any message passing through the network.
  - He is a legitimate user of the network → He can initiate a conversation with any other user.
  - He will have the opportunity to become a receiver to any principal.
  - He can send messages to any principal by impersonating any other principal
Dolev-Yao Threat Model (4)

- Malice **cannot**:
  - He cannot guess a random number which is chosen from a sufficiently large space.
  - Without the correct secret (or private) key, he cannot retrieve plaintext from given ciphertext, and cannot create valid ciphertext from given plaintext, (with respect to the perfect encryption algorithm).
  - He cannot find the private key matching a given public key.
  - While he may have control of a large public part of our computing and communication environment, in general, he is not in control of many private areas of the computing environment, such as accessing the memory of a principal's offline computing device.
Suppose that two principals Alice and Bob wish to communicate with each other in a secure manner.

Suppose that Alice and Bob have never met before, and therefore they do not already share a secret key between them and do not already know for sure the other party's public key.

Then how can they communicate securely over completely insecure networks?
Outline

- Cryptography Overview
- Classic Cryptography
- Modern Cryptography
- Key Channel
- Perfect Encryption
- Dolev-Yao Threat Model
- Key Exchange Protocols
Outline

- Key Exchange Protocols
  - Protocol “From Alice to Bob”
  - Protocol “Session Key from Trent”
  - Protocol “Message Authentication”
  - Protocol “Challenge-response”
  - Protocol using Public-key Cryptosystems.
Protocol “From Alice to Bob” (1)

- **Premise:**
  - Alice and Trent shared key $K_{AT}$.
  - Bob and Trent shared key $K_{BT}$.

- **Goal:** Alice and Bob want to establish a new and share secret key $K$ (new session key) for communication.

- Alice is an initiator. Bob is a responder.

- Alice and Bob know Trent and share respective long-term keys with Trent; so it is possible for Trent to securely pass messages between them.
Protocol “From Alice to Bob” (2)

1. Alice generates a session key $K$ at random; creates ${K}_{K_{AT}}$; sends to Trent: Alice, Bob, ${K}_{K_{AT}}$

2. Trent finds $K_{AT}$, $K_{BT}$; decrypts ${K}_{K_{AT}}$ to reveal $K$, creates ${K}_{K_{BT}}$; sends to Bob: Alice, Bob, ${K}_{K_{BT}}$

3. Bob decrypts ${K}_{K_{BT}}$ to reveal; forms and sends to Alice: 
   {Hello Alice, I'm Bob!}$_K$
Protocol “From Alice to Bob” (3)

- **Problem**: K created by Alice is not strong enough (not sufficiently random or easily memorable)
  - Bob's security can be compromised since the key is a shared one.
  - Bob does not trust Alice and should not feel comfortable accepting a session key generated by her and sharing with her.
- **New protocol**: “Session Key from Trent”
Protocol “Session Key from Trent” (1)

- **Premise:**
  - Alice and Trent shared key $K_{AT}$.
  - Bob and Trent shared key $K_{BT}$.

- **Goal:** Alice and Bob want to establish a new and share secret key $K$ (new session key) for communication.

- In this protocol, the new session key is created by Trent.
1. Alice sends to Trent: \textit{Alice, Bob}

2. Trent finds $K_{AT}$, $K_{BT}$; generates $K$ at random and sends to Alice: 
\textit{\{K\}$_{K_{AT}}$, \{K\}$_{K_{BT}}$}

3. Alice decrypts \{K\}$_{K_{AT}}$; sends to Bob: \textit{Trent, Alice, \{K\}$_{K_{BT}}$}

4. Bob decrypts \{K\}$_{K_{BT}}$ to reveal $K$; forms and sends to Alice: 
\textit{\{Hello Alice, I'm Bob!\}$_{K}$}
Protocol “Session key from Trent” – Attack 1 (1)

1. Alice sends to Malice (“Trent”): Alice, Bob
1’. Malice (“Alice”) sends to Trent: Alice, Malice
2. Trent finds $K_{AT}$, $K_{MT}$; generates $K_{AM}$ at random and sends to Alice: $\{K_{AM}\}_{K_{AT}}$, $\{K_{AM}\}_{K_{MT}}$
3. Alice decrypts $\{K_{AM}\}_{K_{AT}}$; sends to Malice (“Bob”): Trent, Alice, $\{K_{AM}\}_{K_{MT}}$
4. Malice (“Bob”) decrypts $\{K_{AM}\}_{K_{MT}}$ to reveal $K_{AM}$; forms and sends to Alice: $\{Hello\, Alice, I'm\, Bob!\}_{K_{AM}}$

$K_{MT}$: shared key of Malice and Trent.
Protocol “Session key from Trent” – Attack 1 (2)

- **Result of the above attack:**
  - Alice will believe that the protocol has been successfully completed with Bob.
  - Malice masquerade as Bob and learn all the information that Alice intends to send to Bob.
  - Bob will not know that he is requested to run the protocol.

- **What is the problem from this attack?**
  - Malice must be a *legitimate user* known to Trent.
  - Inside attackers are often more of a threat than outsiders.
Realize:
• The above attack works as a result of Malice's alteration of Bob's identity.
• Bob's identity is sent in cleartext.
→ hide Bob's identity.

Modify the protocol at the step 1:

1. Alice sends to Trent: $Alice, \{Bob\}_{K_{AT}}$

• Bob's identity is encrypted under the key shared between Alice and Trent.
• Why is Alice’s identity not encrypted?
1. Alice sends to Trent: $Alice, \{Bob\}_{K_{AT}}$
2. Trent finds $K_{AT}$, decrypts $\{Bob\}_{K_{AT}}$ to know whom Alice wants to create a shared key; Trent finds $K_{BT}$; generates $K$ at random and sends to Alice: $\{K\}_{K_{AT}}, \{K\}_{K_{BT}}$
3. Alice decrypts $\{K\}_{K_{AT}}$; sends to Bob: $Trent, Alice, \{K\}_{K_{BT}}$
4. Bob decrypts $\{K\}_{K_{BT}}$ to reveal $K$; forms and sends to Alice: $\{Hello\ Alice, I'm\ Bob!\}_K$
Protocol “Session key from Trent” – Attack 2

- Attack:

1. Alice sends to Trent: $\textit{Alice, }\{\textit{Bob}\}_{K_{AT}}$
1’. Malice(“Alice”) sends to Trent: $\textit{Alice, }\{\textit{Malice}\}_{K_{AT}}$
2. Trent finds $K_{AT}$, decrypts $\{\textit{Bob}\}_{K_{AT}}$ to know whom Alice wants to create a shared key; Trent finds $K_{BT}$; generates $K$ at random and sends to Alice: $\{K\}_{K_{AT}}, \{K\}_{K_{BT}}$
3. Alice decrypts $\{K\}_{K_{AT}}$; sends to Bob: $\textit{Trent, Alice, }\{K\}_{K_{BT}}$
4. Bob decrypts $\{K\}_{K_{BT}}$ to reveal $K$; forms and sends to Alice: $\{\textit{Hello Alice, I’m Bob!}\}_K$

- Why?

  • Why can Malice create $\{\textit{Malice}\}_{K_{AT}}$?
  • Why does Malice know who is the person Alice wants to make conversation?

- Result: Malice can still successfully masquerade as Bob.
Protocol “Session key from Trent” – Attack 3

- Another attack:
  - In a previous protocol run (a correct run) between Alice and Malice, Malice has recorded $K'$ and the ciphertext part $\{K'\}_{K_{AT}}$.
  - Malice can alter the message from Trent to Alice:
    1. Alice sends to Trent: $Alice, \{Bob\}_{K_{AT}}$
    2'. Malice("Trent") sends to Alice: $\{K'\}_{K_{AT}}, \{K'\}_{K_{MT}}$

- Result: Malice can still successfully masquerade as Bob by using the old session key $K'$. 
Protocol “Session key from Trent” – Problem

- Malice is able to alter some protocol messages without being detected.
- Thus the protocol needs a security service which can guard against tampering of messages.
- Protocol “Message Authentication”
Protocol “Message Authentication” (1)

1. Alice sends to Trent: *Alice, Bob*
2. Trent finds $K_{AT}, K_{BT}$; generates $K$ at random and sends to Alice: 
   ${Bob, K}_{K_{AT}}, {Alice, K}_{K_{BT}}$
3. Alice decrypts ${Bob, K}_{K_{AT}}$, checks Bob’s identity and sends to Bob: 
   *Trent, {Alice, K}$_{K_{BT}}$
4. Bob decrypts ${Alice, K}_{K_{BT}}$, checks Alice’s identity and sends to Alice: 
   *{Hello Alice, I'm Bob!}$_{K}$*
Protocol “Message Authentication” (2)

- Based on the Property 3 of Perfect Encryption:
  Without the key $K$, even with the knowledge of the plaintext $M$, it should be impossible for someone to alter $\{M\}_K$ without being detected by the recipient during the time of decryption.

  $\rightarrow$ Malice cannot modify $\{Bob, K\}_{K_{AT}}, \{Alice, K\}_{K_{BT}}$ without being detected.
Protocol “Message Authentication” - Attack

- Problem: message replay attack.
- Malice intercepts Alice's request, then:
  1. Alice sends to Malice(“Trent”)
  2’. Malice("Trent") sends to Alice: \(\{Bob, K'\}_K_{AT}, \{Alice, K'\}_K_{BT}\)
- 2 ciphertext blocks containing \(K'\) are a replay of old messages which Malice has recorded from a previous run of the protocol (a normal run between Alice and Bob).
  → This attack will cause Alice and Bob to reuse the old session key \(K'\).
- Since \(K'\) is old, it may be possible for Malice to have discovered its value (HOW ?? → homework).
Protocol “Challenge-response”

- Giao thức “Challenge-response” bổ sung thêm một số bước nhằm giúp cho Alice và Bob xác nhận một khóa phiên có mới hay không.
  → Chống lại message replay attack
- Giao thức này được Needham và Schroeder đề nghị năm 1978 và còn được gọi là giao thức “Needham and Schroeder”
- Giao thức sử dụng số Nonce (a number used once) – số chỉ được sử dụng 1 lần
- Symmetric-key Authentication Protocol
- Needham and Schroeder which they published in 1978
- Nonce: a number used once
Giao thức “Challenge-response”

1. Alice tạo số $N_A$ ngẫu nhiên và gửi cho Trent: $Alice, Bob, N_A$
2. Trent tạo khóa $K$ ngẫu nhiên và gửi cho Alice: ${N_A, K, Bob, \{K, Alice\}_{KBT}}_{KAT}$
3. Alice giải mã, kiểm tra số $N_A$, kiểm tra danh định của Bob, và gửi cho Bob: $Trent, \{K, Alice\}_{KBT}$
4. Bob giải mã, kiểm tra danh định của Alice, tạo số $N_B$ ngẫu nhiên và gửi cho Alice: $\{I'm Bob! N_B\}_K$
5. Alice gửi cho Bob: $\{I'm Alice! N_B-1\}_K$

$N_A/N_B$: số Nonce tạo bởi Alice/Bob
Tận công giao thức “Challenge-response”

1. Alice gửi cho Trent: Alice, Bob, \( N_A \)
2. Trent gửi cho Alice: \( \{N_A, K, Bob, \{K, Alice\}\}_{K_{BT}} \)
3. Alice gửi cho Malice(“Bob”): Trent, \( \{K, Alice\}\)_{K_{BT}}
3’. Malice(“Alice”) gửi cho Bob: Trent, \( \{K’, Alice\}\)_{K_{BT}}
4. Bob giải mã, kiểm tra danh định của Alice, tạo số \( N_B \) ngẫu nhiên và gửi cho Malice(“Alice”): \( \{‘I’m Bob! N_B\}\)_{K’}
5. Malice(“Alice”) gửi cho Bob: \( \{‘I’m Alice!N_B-1\}\)_{K’}
Tấn công giao thức “Challenge-response”

- Kết quả của tấn công này là:
  - Bob nghĩ rằng mình đang trao đổi một khóa phiên mới với Alice nhưng thật ra đây là một khóa phiên cũ và Malice có thể biết khóa cũ này.
  - Malice đóng giả Alice để nói chuyện với Bob
  - Alice không thực hiện thành công cuộc nói chuyện với Bob
Giao thức “Challenge-response”

- Alice dựa vào số $N_A$ để xác định thông điệp được dùng là do Trent gửi và khóa phiên là mới (bước 1-2-3)
- Bob không có cơ sở để xác định khóa phiên là do Trent tạo ra và là khóa phiên mới
  → Bổ sung thêm một số thông điệp trao đổi giữa Bob và Trent
  → Sử dụng Timestamps
Giao thức “Challenge-response” với Timestamps

1. Alice gửi cho Trent: *Alice, Bob*
2. Trent gửi cho Alice: \{Bob, K, T, \{Alice, K, T\}_{KBT}\}_{KAT}
3. Alice kiểm tra T và gửi cho Bob: \{Alice, K, T\}_{KBT}
4. Bob kiểm tra T và gửi cho Alice: \{I’m Bob! N_{B}\}_K
5. Alice gửi cho Bob: \{I’m Alice!N_{B-1}\}_K

- Kiểm tra T: \|\text{Clock} – T\| < \Delta t_1 + \Delta t_2
  - Clock: đồng hồ tại máy cá nhân
  - T: timestamp, giờ tại Trent
  - \Delta t_1, \Delta t_2: độ lệch múi giờ và độ lệch thời gian cho phép

Không được áp dụng do khó có thể điều chỉnh giờ chuẩn rộng rãi.
Giao thức dùng Mã hóa công khai

- $K_A, K^{-1}_A$: là khóa công khai và khóa bí mật của Alice
- $K_B, K^{-1}_B$: là khóa công khai và khóa bí mật của Bob
- $K_M, K^{-1}_M$: là khóa công khai và khóa bí mật của Malice
- $K_T, K^{-1}_T$: là khóa công khai và khóa bí mật của Trent
- $\{M\}_{K_A}$: mã hóa $M$ bằng khóa công khai $K_A$
- $\{M\}_{K^{-1}_A}$: ký lên $M$ bằng khóa bí mật $K_A$
1. Alice gửi cho Trent: **Alice, Bob**
2. Trent gửi cho Alice: \( \{K_B, Bob\}_{K^{-1}_T} \)
3. Alice kiểm tra chữ ký của Trent, tạo số \( N_A \) và gửi cho Bob: \( \{N_A, Alice\}_{KB} \)
4. Bob giải mã, kiểm tra danh định của Alice và gửi cho Trent: **Bob, Alice**
5. Trent gửi cho Bob: \( \{K_A, Alice\}_{K^{-1}_T} \)
6. Bob kiểm tra chữ ký của Trent, tạo số \( N_B \) và gửi cho Alice: \( \{N_A, N_B\}_{KA} \)
7. Alice giải mã và gửi cho Bob: \( \{N_B\}_{KB} \)
Giao thức dùng Mã hóa công khai

- Kết quả của giao thức là Alice và Bob cùng có chung hai số nonce $N_A$ và $N_B$. Khóa chung bí mật được tạo thành từ 2 số này.
- Nhưng ...

... vẫn có cách tấn công giao thức này

- Được khám phá sau 17 năm
- Cách tấn công: Malice lợi dụng lúc Alice muốn nói chuyện với mình để giả mạo Alice
Tấn công giao thức dùng Mã hóa công khai

Giao thức giữa Alice & Malice

Giao thức giữa Malice (“Alice”) & Bob
Tấn công giao thức dùng Mã hóa công khai

- Kết quả của tấn công này là:
  - Bob nghĩ rằng mình đang trao đổi 2 số nonce bí mật NA, NB với Alice nhưng thật ra là với Malice
  - Alice và Malice vẫn có cuộc nói chuyện bình thường.

- Ví dụ: Nếu Bob là một ngân hàng, Malice ("Alice") gửi cho Bob một yêu cầu sau:

  \[ \{N_A, N_B, \text{"Transfer £1B from my account to Malice's"}\}_K_B \]
Tấn công giao thức dùng Mã hóa công khai

- Cách khắc phục: kết hợp thêm chữ ký điện tử từ lên NA, NB
- Giao thức được sử dụng hiện nay là:
  - Gọi chung cặp khóa bí mật và khóa công khai của Alice là $K_A$
  - $\{M\}_K$: mã hóa $M$ bằng khóa $K$
  - $[M]_K$: chữ ký điện tử từ lên $M$ bằng khóa $K$

The Needham-Schroeder Public-key Authentication Protocol in Refined Specification

1. Alice sends to Bob: $\{[NA, Alice]_{KA})_{KB}$
2. Bob sends to Alice: $\{NA, [NB]_{KB})_{KA}$
3. Alice sends to Bob: $\{[NB]_{KA})_{KB}$
Outline

- **Key Exchange Protocols**
  - Protocol “From Alice to Bob”
  - Protocol “Session Key from Trent”
  - Protocol “Message Authentication”
  - Protocol “Challenge-response”
  - Protocol using Public-key Cryptosystems.