Symetric Encryption et al.

Pascal Lafourcade

Université Joseph Fourier, Verimag

7th October 2009
Last Time

- Presentation
- Motivation
- History of Cryptography
- Classical Asymmetric Encryption
Outline

Classical Symmetric Encryptions
  DES
  3-DES
  AES
  IDEA
Modes
  ECB
  CBC
  CFB
  OFB
Asymmetric vs Symetric
  Diffie-Hellman
Hash Functions
Applications
Conclusion
Outline

Classical Symmetric Encryptions
  DES
  3-DES
  AES
  IDEA

Modes
  ECB
  CBC
  CFB
  OFB

Asymmetric vs Symmetric

Diffie-Hellman

Hash Functions

Applications

Conclusion
**Data Encryption Standard**, (call in 1973)

Lucifer designed in 1971 by Horst Feistel at IBM.

- Block cipher, encrypting 64-bit blocks
  - Uses 56 bit keys
  - Expressed as 64 bit numbers (8 bits parity checking)

  ![Diagram](image)

- First cryptographic standard.
  - 1977 US federal standard (US Bureau of Standards)
  - 1981 ANSI private sector standard
DES — overall form

- 16 rounds Feistel cipher + key-scheduler.
- Key scheduling algorithm derives subkeys $K_i$ from original key $K$.
- Initial permutation at start, and inverse permutation at end.
- $f$ consists of two permutations and an s-box substitution.

$L_{i+1} = R_i$ and $R_{i+1} = L_i \oplus f(R_i, K_i)$
DES — 1 round

\[(b_1 b_6, b_2 b_3 b_4 b_5), C_j \text{ represents the binary value in the row } b_1 b_6 \text{ and column } b_2 b_3 b_4 b_5 \text{ of the } S_j \text{ box.}\]
Symmetric Encryption et al.
Classical Symmetric Encryptions

**DES**

**S-Boxes: S1, S2, S3, S4**

<table>
<thead>
<tr>
<th>14</th>
<th>4</th>
<th>13</th>
<th>1</th>
<th>2</th>
<th>15</th>
<th>11</th>
<th>8</th>
<th>3</th>
<th>10</th>
<th>6</th>
<th>12</th>
<th>5</th>
<th>9</th>
<th>0</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>15</td>
<td>7</td>
<td>4</td>
<td>14</td>
<td>2</td>
<td>13</td>
<td>1</td>
<td>10</td>
<td>6</td>
<td>12</td>
<td>11</td>
<td>9</td>
<td>5</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>14</td>
<td>8</td>
<td>13</td>
<td>6</td>
<td>2</td>
<td>11</td>
<td>15</td>
<td>12</td>
<td>9</td>
<td>7</td>
<td>3</td>
<td>10</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>12</td>
<td>8</td>
<td>2</td>
<td>4</td>
<td>9</td>
<td>1</td>
<td>7</td>
<td>5</td>
<td>11</td>
<td>3</td>
<td>14</td>
<td>10</td>
<td>0</td>
<td>6</td>
<td>13</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>1</th>
<th>8</th>
<th>14</th>
<th>6</th>
<th>11</th>
<th>3</th>
<th>4</th>
<th>9</th>
<th>7</th>
<th>2</th>
<th>13</th>
<th>12</th>
<th>0</th>
<th>5</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>13</td>
<td>4</td>
<td>7</td>
<td>15</td>
<td>2</td>
<td>8</td>
<td>14</td>
<td>12</td>
<td>0</td>
<td>1</td>
<td>10</td>
<td>6</td>
<td>9</td>
<td>11</td>
<td>5</td>
</tr>
<tr>
<td>0</td>
<td>14</td>
<td>7</td>
<td>11</td>
<td>10</td>
<td>4</td>
<td>13</td>
<td>1</td>
<td>5</td>
<td>8</td>
<td>12</td>
<td>6</td>
<td>9</td>
<td>3</td>
<td>2</td>
<td>15</td>
</tr>
<tr>
<td>13</td>
<td>8</td>
<td>10</td>
<td>1</td>
<td>3</td>
<td>15</td>
<td>4</td>
<td>2</td>
<td>11</td>
<td>6</td>
<td>7</td>
<td>12</td>
<td>0</td>
<td>5</td>
<td>14</td>
<td>9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>10</th>
<th>0</th>
<th>9</th>
<th>14</th>
<th>6</th>
<th>3</th>
<th>15</th>
<th>5</th>
<th>1</th>
<th>13</th>
<th>12</th>
<th>7</th>
<th>11</th>
<th>4</th>
<th>2</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>7</td>
<td>0</td>
<td>9</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>10</td>
<td>2</td>
<td>8</td>
<td>5</td>
<td>14</td>
<td>12</td>
<td>11</td>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>6</td>
<td>4</td>
<td>9</td>
<td>8</td>
<td>15</td>
<td>3</td>
<td>0</td>
<td>11</td>
<td>1</td>
<td>2</td>
<td>12</td>
<td>5</td>
<td>10</td>
<td>14</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>13</td>
<td>0</td>
<td>6</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>4</td>
<td>15</td>
<td>14</td>
<td>3</td>
<td>11</td>
<td>5</td>
<td>2</td>
<td>12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>7</th>
<th>13</th>
<th>14</th>
<th>3</th>
<th>0</th>
<th>6</th>
<th>9</th>
<th>10</th>
<th>1</th>
<th>2</th>
<th>8</th>
<th>5</th>
<th>11</th>
<th>12</th>
<th>4</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>8</td>
<td>11</td>
<td>5</td>
<td>6</td>
<td>15</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>7</td>
<td>2</td>
<td>12</td>
<td>1</td>
<td>10</td>
<td>14</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>6</td>
<td>9</td>
<td>0</td>
<td>12</td>
<td>11</td>
<td>7</td>
<td>13</td>
<td>15</td>
<td>1</td>
<td>3</td>
<td>14</td>
<td>5</td>
<td>2</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>0</td>
<td>6</td>
<td>10</td>
<td>1</td>
<td>13</td>
<td>8</td>
<td>9</td>
<td>4</td>
<td>5</td>
<td>11</td>
<td>12</td>
<td>7</td>
<td>2</td>
<td>14</td>
</tr>
</tbody>
</table>
## S-Boxes: S5, S6, S7 and S8

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>12</th>
<th>4</th>
<th>1</th>
<th>7</th>
<th>10</th>
<th>11</th>
<th>6</th>
<th>8</th>
<th>5</th>
<th>3</th>
<th>15</th>
<th>13</th>
<th>0</th>
<th>14</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>11</td>
<td>2</td>
<td>12</td>
<td>4</td>
<td>7</td>
<td>13</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>15</td>
<td>10</td>
<td>3</td>
<td>9</td>
<td>8</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>1</td>
<td>11</td>
<td>10</td>
<td>13</td>
<td>7</td>
<td>8</td>
<td>15</td>
<td>9</td>
<td>12</td>
<td>5</td>
<td>6</td>
<td>3</td>
<td>0</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>8</td>
<td>12</td>
<td>7</td>
<td>1</td>
<td>14</td>
<td>2</td>
<td>13</td>
<td>6</td>
<td>15</td>
<td>0</td>
<td>9</td>
<td>10</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>12</th>
<th>1</th>
<th>10</th>
<th>15</th>
<th>9</th>
<th>2</th>
<th>6</th>
<th>8</th>
<th>0</th>
<th>13</th>
<th>3</th>
<th>4</th>
<th>14</th>
<th>7</th>
<th>5</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>15</td>
<td>4</td>
<td>2</td>
<td>7</td>
<td>12</td>
<td>9</td>
<td>5</td>
<td>6</td>
<td>1</td>
<td>13</td>
<td>14</td>
<td>0</td>
<td>11</td>
<td>3</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>14</td>
<td>15</td>
<td>5</td>
<td>2</td>
<td>8</td>
<td>12</td>
<td>3</td>
<td>7</td>
<td>0</td>
<td>4</td>
<td>10</td>
<td>1</td>
<td>13</td>
<td>11</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td>12</td>
<td>9</td>
<td>5</td>
<td>15</td>
<td>10</td>
<td>11</td>
<td>14</td>
<td>1</td>
<td>7</td>
<td>6</td>
<td>0</td>
<td>8</td>
<td>13</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>4</th>
<th>11</th>
<th>2</th>
<th>14</th>
<th>15</th>
<th>0</th>
<th>8</th>
<th>13</th>
<th>3</th>
<th>12</th>
<th>9</th>
<th>7</th>
<th>5</th>
<th>10</th>
<th>6</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>0</td>
<td>11</td>
<td>7</td>
<td>4</td>
<td>9</td>
<td>1</td>
<td>10</td>
<td>14</td>
<td>3</td>
<td>5</td>
<td>12</td>
<td>2</td>
<td>15</td>
<td>8</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>11</td>
<td>13</td>
<td>12</td>
<td>3</td>
<td>7</td>
<td>14</td>
<td>10</td>
<td>15</td>
<td>6</td>
<td>8</td>
<td>0</td>
<td>5</td>
<td>9</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>11</td>
<td>13</td>
<td>8</td>
<td>1</td>
<td>4</td>
<td>10</td>
<td>7</td>
<td>9</td>
<td>5</td>
<td>0</td>
<td>15</td>
<td>14</td>
<td>2</td>
<td>3</td>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>13</th>
<th>2</th>
<th>8</th>
<th>4</th>
<th>6</th>
<th>15</th>
<th>11</th>
<th>1</th>
<th>10</th>
<th>9</th>
<th>3</th>
<th>14</th>
<th>5</th>
<th>0</th>
<th>12</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15</td>
<td>13</td>
<td>8</td>
<td>10</td>
<td>3</td>
<td>7</td>
<td>4</td>
<td>12</td>
<td>5</td>
<td>6</td>
<td>11</td>
<td>0</td>
<td>14</td>
<td>9</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>11</td>
<td>4</td>
<td>1</td>
<td>9</td>
<td>12</td>
<td>14</td>
<td>2</td>
<td>0</td>
<td>6</td>
<td>10</td>
<td>13</td>
<td>15</td>
<td>3</td>
<td>5</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>14</td>
<td>7</td>
<td>4</td>
<td>10</td>
<td>8</td>
<td>13</td>
<td>15</td>
<td>12</td>
<td>9</td>
<td>0</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>11</td>
<td></td>
</tr>
</tbody>
</table>
Decryption DES
Use inverse sequence key.

- $IP(C) = IP(IP^{-1}(R_{16} || L_{16}))$
- $L'_0 = R_{16}$ and $R'_0 = L_{16}$

\[
L'_1 = R'_0 = L_{16} = R_{15}
\]

\[
R'_1 = L'_0 \oplus f(R'_0, K'_0)
\]

\[
R'_1 = R_{16} \oplus f(L_{16}, K_{15})
\]

\[
R'_1 = R_{16} \oplus f(R_{15}, K_{15})
\]

\[
R'_1 = L_{15}
\]

Recall $L_{i+1} = R_i$ and $R_{i+1} = L_i \oplus f(R_i, K_i)$
Property of DES DES exhibits the complementation property, namely that

\[ E_K(P) = C \iff E_K(P) = \overline{C} \]

where \( \overline{x} \) is the bitwise complement of \( x \). \( E_K \) denotes encryption with key \( K \). Then \( P \) and \( C \) denote plaintext and ciphertext blocks respectively.
Anomalies of DES

- Existence of 6 pairs of semi-weak keys: $E_{k_1}(E_{k_2}(x)) = x$.
  - $0x011F011F010E010E$ and $0x1F011F010E010E$
  - $0x01E001E001F101F1$ and $0xE001E001F101F1$
  - $0x01FE01FE01FE01FE$ and $0xFE01FE01FE01FE$
  - $0x1FE01FE00EF10EF1$ and $0xE01FE01FF10EF1$
  - $0x1FFE1FFE0EFE0EFE$ and $0xFE1FFE1FFE0EFE$
  - $0xE0FEE0FEF1FEF1FE$ and $0xFEE0FEE0FEF1FEF1$
Security of DES

- No security proofs or reductions known
- Main attack: exhaustive search
  - 7 hours with 1 million dollar computer (in 1993).
  - 7 days with $10,000 FPGA-based machine (in 2006).
- Mathematical attacks
  - Not know yet.
  - But it is possible to reduce key space from $2^{56}$ to $2^{43}$ using (linear) cryptanalysis.
    - To break the full 16 rounds, differential cryptanalysis requires $2^{47}$ chosen plaintexts (Eli Biham and Adi Shamir).
    - Linear cryptanalysis needs $2^{43}$ known plaintexts (Matsui, 1993)
Triple DES

- Use three stages of encryption instead of two.

- Compatibility is maintained with standard DES ($K_2 = K_1$).
- No known practical attack
  $\Rightarrow$ brute-force search with $2^{112}$ operations.
Advanced Encryption Standard

- Block-size = 128 bits, Key size = 128, 192, or 256 bits.
- Uses various substitutions and transpositions + key scheduling, in different rounds.
- Algorithm believed secure. Only attacks are based on side channel analysis, i.e., attacking implementations that inadvertently leak information about the key.

<table>
<thead>
<tr>
<th>Key Size</th>
<th>Round Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>10</td>
</tr>
<tr>
<td>192</td>
<td>12</td>
</tr>
<tr>
<td>256</td>
<td>14</td>
</tr>
</tbody>
</table>
AES: High-level cipher algorithm

- KeyExpansion using Rijndael’s key schedule
- Initial Round: AddRoundKey
- Rounds:
  1. SubBytes: a non-linear substitution step where each byte is replaced with another according to a lookup table.
  2. ShiftRows: a transposition step where each row of the state is shifted cyclically a certain number of steps.
  3. MixColumns: a mixing operation which operates on the columns of the state, combining the four bytes in each column
  4. AddRoundKey: each byte of the state is combined with the round key; each round key is derived from the cipher key using a key schedule.

- Final Round (no MixColumns)
  1. SubBytes
  2. ShiftRows
  3. AddRoundKey
AES: SubBytes

SubBytes: a non-linear substitution step where each byte is replaced with another according to a lookup table.
AES: ShiftRows

ShiftRows: a transposition step where each row of the state is shifted cyclically a certain number of steps.
AES: MixColumns

MixColumns: a mixing operation which operates on the columns of the state, combining the four bytes in each column
**AES: AddRoundKey**

AddRoundKey: each byte of the state is combined with the round key; each round key is derived from the cipher key using a key
AES: Attacks

Not yet efficient Cryptanalysis on complete version, but Niels Ferguson proposed in 2000 an attack on a version with 7 rounds and 128 bits key.

But

Marine Minier, Raphael C.-W. Phan, Benjamin Pousse: Distinguishers for Ciphers and Known Key Attack against Rijndael with Large Blocks. AFRICACRYPT 2009: 60-76

Samuel Galice, Marine Minier: Improving Integral Attacks Against Rijndael-256 Up to 9 Rounds. AFRICACRYPT 2008: 1-15

Side channel attacks using on optimized version (2005)

- Timing.
- Cache Default.
- Electric Consumptions.
- ..

There exists algebraic attacks ...
IDEA: International Data Encryption Algorithm 1991

Designed by Xuejia Lai and James Massey of ETH Zurich.
IDEA uses a message of 64-bit blocks and a 128-bit key,

**Key schedule**

- K1 to K6 for the first round are taken directly as the first 6 consecutive blocks of 16 bits.
- This means that only 96 of the 128 bits are used in each round.
- 128 bit key undergoes a 25 bit rotation to the left, i.e. the LSB becomes the 25th LSB.
### IDEA

#### Notation

- Bitwise eXclusive OR (denoted with a blue \( \oplus \)).
- Addition modulo 216 (denoted with a green \( \boxplus \)).
- Multiplication modulo 216+1, where the all-zero word (0x0000) is interpreted as 216 (denoted by a red \( \odot \)).
Symmetric Encryption et al.
Classical Symmetric Encryptions
IDEA

IDEA

[Diagram of IDEA encryption process]
After the eight rounds comes a final "half round".
After the eight rounds comes a final "half round".

The best attack which applies to all keys can break IDEA reduced to 6 rounds (the full IDEA cipher uses 8.5 rounds). Biham, E. and Dunkelman, O. and Keller, N. "A New Attack on 6-Round IDEA".
Others Symmetric Encryption Schemes

Outline

Classical Symmetric Encryptions
  DES
  3-DES
  AES
  IDEA

Modes
  ECB
  CBC
  CFB
  OFB

Asymmetric vs Symmetric
Diffie-Hellman
Hash Functions
Applications
Conclusion
Electronic Book Code (ECB)

Each block of the same length is encrypted separately using the same key $K$. In this mode, only the block in which the flipped bit is contained is changed. Other blocks are not affected.
ECB Encryption Algorithm

```plaintext
algorithm \( E_K(M) \)
if (\(|M| \mod n \neq 0\) or \(|M| = 0\)) then return FAIL
Break \( M \) into n-bit blocks \( M[1] \ldots M[m] \)
for \( i = 1 \) to \( m \) do \( C[i] = E_K(M[i]) \)
\( C = C[1] \ldots C[m] \)
return \( C \)
```
Electronic Codebook (ECB) mode encryption
ECB Decryption Algorithm

**algorithm** $D_K(C)$

if ($|C| \mod n \neq 0$ or $|C| = 0$) then return FAIL

Break $C$ into $n$-bit blocks $C[1] \ldots C[m]$

for $i = 1$ to $m$ do $M[i] = D_K(C[i])$

$M = M[1] \ldots M[m]$

return $M$
Electronic Codebook (ECB) mode decryption
Cipher-block chaining (CBC)

If the first block has index 1, the mathematical formula for CBC encryption is

\[ C_i = E_K(P_i \oplus C_{i-1}), \quad C_0 = IV \]

while the mathematical formula for CBC decryption is

\[ P_i = D_K(C_i) \oplus C_{i-1}, \quad C_0 = IV \]

CBC has been the most commonly used mode of operation.
Cipher Block Chaining (CBC) mode encryption
Symmetric Encryption et al.
Modes
CBC

Cipher Block Chaining (CBC) mode decryption
The cipher feedback (CFB)

A close relative of CBC:

\[ C_i = E_K(C_{i-1}) \oplus P_i \]

\[ P_i = E_K(C_{i-1}) \oplus C_i \]

\[ C_0 = IV \]
Cipher Feedback (CFB) mode encryption
Cipher Feedback (CFB) mode decryption
Output feedback (OFB)

Because of the symmetry of the XOR operation, encryption and decryption are exactly the same:

\[ C_i = P_i \oplus O_i \]

\[ P_i = C_i \oplus O_i \]

\[ O_i = E_K(O_{i-1}) \]

\[ O_0 = IV \]
Symmetric Encryption et al.

Modes

OFB

Initialization Vector (IV)

Key → Block Cipher Encryption → Ciphertext

Key → Block Cipher Encryption → Ciphertext

Key → Block Cipher Encryption → Ciphertext

Output Feedback (OFB) mode encryption
Output Feedback (OFB) mode decryption
ECB vs Others
Outline

Classical Symmetric Encryptions
  DES
  3-DES
  AES
  IDEA

Modes
  ECB
  CBC
  CFB
  OFB

Asymmetric vs Symetric
  Diffie-Hellman
  Hash Functions
  Applications
  Conclusion
Comparison

- Size of the key
- Complexity of computation (time, hardware, cost ...)
- Number of different keys?
- Key distribution
- Signature only possible with asymmetric scheme
## Computational cost of encryption

**2 hours of video (assumes 3Ghz CPU)**

<table>
<thead>
<tr>
<th>Schemes</th>
<th>Encrypt</th>
<th>Decrypt</th>
<th>Encrypt</th>
<th>Decrypt</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSA 2048(1)</td>
<td>22 min</td>
<td>24 h</td>
<td>115 min</td>
<td>130 h</td>
</tr>
<tr>
<td>RSA 1024(1)</td>
<td>21 min</td>
<td>10 h</td>
<td>111 min</td>
<td>53 h</td>
</tr>
<tr>
<td>AES CTR(2)</td>
<td>20 sec</td>
<td>20 sec</td>
<td>105 sec</td>
<td>105 sec</td>
</tr>
</tbody>
</table>
Outline

Classical Symetric Encryptions
  DES
  3-DES
  AES
  IDEA

Modes
  ECB
  CBC
  CFB
  OFB

Asymmetric vs Symetric

Diffie-Hellman

Hash Functions

Applications

Conclusion
The Diffie-Hellman protocol

\[ g, p \] are public parameters.

Diffie chooses \( x \) and computes \( g^x \mod p \).
Hellman chooses \( y \) and computes \( g^y \mod p \).

**Basic Diffie-Hellman key-exchange**: initiator I and responder R exchange public “half-keys” to arrive at mutual session key
\[ k = g^{xy} \mod p. \]
The Diffie-Hellman protocol

$g, p$ are public parameters.

Diffie chooses $x$ and computes $g^x \mod p$.  
Hellman chooses $y$ and computes $g^y \mod p$.  

**Basic Diffie-Hellman key-exchange**: initiator I and responder R exchange public “half-keys” to arrive at mutual session key $k = g^{xy} \mod p$.  

$$g^x \mod p \quad \rightarrow \quad g^y \mod p$$
Hard Problems

Most cryptographic constructions are based on *hard problems*. Their security is proved by reduction to these problems:

- **RSA.** Given \( N = pq \) and \( e \in \mathbb{Z}^*_\varphi(N) \), compute the inverse of \( e \) modulo \( \varphi(N) = (p - 1)(q - 1) \). Factorization
- **Discrete Logarithm problem, DL.** Given a group \( \langle g \rangle \) and \( g^x \), compute \( x \).
- **Computational Diffie-Hellman, CDH** Given a group \( \langle g \rangle \), \( g^x \) and \( g^y \), compute \( g^{xy} \).
- **Decisional Diffie-Hellman, DDH** Given a group \( \langle g \rangle \), distinguish between the distributions \((g^x, g^y, g^{xy})\) and \((g^x, g^y, g^r)\).
The Discrete Logarithm (DL)

Let $G = (\langle g \rangle, \ast)$ be any finite cyclic group of prime order. Idea: it is hard for any adversary to produce $x$ if he only knows $g^x$. For any adversary $\mathcal{A}$,

$$\text{Adv}^{DL}(\mathcal{A}) = Pr\left[\mathcal{A}(g^x) \rightarrow x \mid x, y \xleftarrow{R} [1, q]\right]$$

is negligible.
Computational Diffie-Hellman (CDH)

Idea: it is hard for any adversary to produce $g^{xy}$ if he only knows $g^x$ and $g^y$.
For any adversary $A$,

$$\text{Adv}^{\text{CDH}}(A) = \Pr[A(g^x, g^y) \rightarrow g^{xy} \mid x, y \leftarrow [1, q]]$$

is negligible.
Decisional Diffie-Hellman (DDH)

Idea: Knowing $g^x$ and $g^y$, it should be hard for any adversary to distinguish between $g^{xy}$ and $g^r$ for some random value $r$. For any adversary $\mathcal{A}$, the advantage of $\mathcal{A}$

$$\text{Adv}^{\text{DDH}}(\mathcal{A}) = \Pr \left[ \mathcal{A}(g^x, g^y, g^{xy}) \to 1 \mid x, y \overset{R}{\leftarrow} [1, q] \right]$$

$$- \Pr \left[ \mathcal{A}(g^x, g^y, g^r) \to 1 \mid x, y, r \overset{R}{\leftarrow} [1, q] \right]$$

is negligible.
This means that an adversary cannot extract a single bit of information on $g^{xy}$ from $g^x$ and $g^y$. 
Relation between the problems

Prop

Solve $DL \Rightarrow$ Solve $CDH \Rightarrow$ Solve $DDH$. (Exercise)

Prop (Moaurer & Wolf)

For many groups, $DL \Leftrightarrow CDH$

Prop (Joux & Wolf)

There are groups for which $DDH$ is easier than $CDH$. 
Proofs by Reduction

Solve DL $\Rightarrow$ Solve CDH

Attack on DL implies attack on CDH.
Given $g, g^x, g^y$ using DL we get $x$ and $y$ so we can compute $g^{xy}$.

Solve CDH $\Rightarrow$ Solve DDH

Attack on CDH implies attack on DDH.
Given $g, g^x, g^y, g^r$ using CDH we compute $g^{xy}$ and we can compare with $g^r$. 
Usage of DH assumption

The Diffie-Hellman problems are widely used in cryptography:

- Public key crypto-systems [ElGamal, Cramer & Shoup]
- Pseudo-random functions [Noar & Reingold, Canetti]
- Pseudo-random generators [Blum & Micali]
- (Group) key exchange protocols [many]
Outline

Classical Symmetric Encryptions
- DES
- 3-DES
- AES
- IDEA

Modes
- ECB
- CBC
- CFB
- OFB

Asymmetric vs Symmetric

Diffie-Hellman

Hash Functions

Applications

Conclusion
### “Classifications” of Hash Functions

<table>
<thead>
<tr>
<th>Unkeyed Hash function</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ Modification Code Detection (MDC)</td>
</tr>
<tr>
<td>▶ Data integrity</td>
</tr>
<tr>
<td>▶ Fingerprints of messages</td>
</tr>
<tr>
<td>▶ Other applications</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Keyed Hash function</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ Message Authentication Code (MAC)</td>
</tr>
<tr>
<td>▶ Password Verification in uncrypted password-image files.</td>
</tr>
<tr>
<td>▶ Key confirmation or establishment</td>
</tr>
<tr>
<td>▶ Time-stamping</td>
</tr>
<tr>
<td>▶ Others applications</td>
</tr>
</tbody>
</table>
Hash Functions

A hash function $\mathcal{H}$ takes as input a bit-string of any finite length and returns a corresponding 'digest' of fixed length.

$$h : \{0, 1\}^* \rightarrow \{0, 1\}^n$$

**Definition (Pre-image resistance (One-way) OWHF)**

Given an output $y$, it is computationally infeasible to compute $x$ such that

$$h(x) = y$$
Properties of hash functions

2nd Pre-image resistance (weak-collision resistant) CRHF

Given an input $x$, it is computationally infeasible to compute $x'$ such that

$$h(x') = h(x)$$

Collision resistance (strong-collision resistant)

It is computationally infeasible to compute $x$ and $x'$ such that

$$h(x) = h(x')$$
Exercises on properties

**collision resistance $\Rightarrow$ 2nd pre-image resistance**

If $h$ is not 2nd pre-image resistance then given $x$ it is possible to compute $x'$ such that

$$h(x') = h(x)$$

which contradict the definition of collision resistance.

But collision resistance does not implies pre-image resistance. Example let $g$ be collision resistant we build $h$ such that

$$h(x) = \begin{cases} 1||x & \text{if } x \text{ has bit length } n \\ 0||g(x) & \text{otherwise} \end{cases}$$

$h$ is collision resistance but not pre-image resistant.
Basic construction of hash functions
Basic construction of hash functions

original input $x$

hash function $h$

preprocessing

append padding bits

append length block

formatted input $x = x_1 x_2 \cdots x_t$

iterated processing

compression function $f$

$H_{i-1}$

$x_i$

$f$

$H_i$

$H_0 = IV$

$g$

output $h(x) = g(H_t)$
Basic construction of hash functions (Merkle-Damgard)

\[ f : \{0, 1\}^m \rightarrow \{0, 1\}^n \]

1. Break the message \( x \) to hash in blocks of size \( m - n \):

\[ x = x_1 x_2 \ldots x_t \]

2. Pad \( x_t \) with zeros as necessary.

3. Define \( x_{t+1} \) as the binary representation of the bit length of \( x \).

4. Iterate over the blocks:

\[ H_0 = 0^n \]

\[ H_i = f(H_{i-1}||x_i) \]

\[ h(x) = H_{t+1} \]
Basic construction of hash functions

**Theorem**

*If the compression function $f$ is collision resistant, then the obtained hash function $h$ is collision resistant.*
Hash functions based on (MDC) block ciphers

Matyas-Meyer-Oseas

Davies-Meyer

Miyaguchi-Preneel
MAC based on block ciphers
## List of Hash Functions

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Output size</th>
<th>Internal state size</th>
<th>Block size</th>
<th>Length size</th>
<th>Word size</th>
<th>Collision</th>
</tr>
</thead>
<tbody>
<tr>
<td>HAVAL</td>
<td>256/.../128</td>
<td>256</td>
<td>1024</td>
<td>64</td>
<td>32</td>
<td>Yes</td>
</tr>
<tr>
<td>MD2</td>
<td>128</td>
<td>384</td>
<td>128</td>
<td>No</td>
<td>8</td>
<td>Almost</td>
</tr>
<tr>
<td>MD4</td>
<td>128</td>
<td>128</td>
<td>512</td>
<td>64</td>
<td>32</td>
<td>Yes</td>
</tr>
<tr>
<td>MD5</td>
<td>128</td>
<td>128</td>
<td>512</td>
<td>64</td>
<td>32</td>
<td>Yes</td>
</tr>
<tr>
<td>PANAMA</td>
<td>256</td>
<td>8736</td>
<td>256</td>
<td>No</td>
<td>32</td>
<td>Yes</td>
</tr>
<tr>
<td>RadioGatn</td>
<td>Arbitrarily long</td>
<td>58 words</td>
<td>3 words</td>
<td>No</td>
<td>1-64</td>
<td>No</td>
</tr>
<tr>
<td>RIPEMD</td>
<td>128</td>
<td>128</td>
<td>512</td>
<td>64</td>
<td>32</td>
<td>Yes</td>
</tr>
<tr>
<td>RIPEMD</td>
<td>128/256</td>
<td>128/256</td>
<td>512</td>
<td>64</td>
<td>32</td>
<td>No</td>
</tr>
<tr>
<td>RIPEMD</td>
<td>160/320</td>
<td>160/320</td>
<td>512</td>
<td>64</td>
<td>32</td>
<td>No</td>
</tr>
<tr>
<td>SHA-0</td>
<td>160</td>
<td>160</td>
<td>512</td>
<td>64</td>
<td>32</td>
<td>Yes</td>
</tr>
<tr>
<td>SHA-1</td>
<td>160</td>
<td>160</td>
<td>512</td>
<td>64</td>
<td>32</td>
<td>With flaws</td>
</tr>
<tr>
<td>SHA-256/224</td>
<td>256/224</td>
<td>256</td>
<td>512</td>
<td>64</td>
<td>32</td>
<td>No</td>
</tr>
<tr>
<td>SHA-512/384</td>
<td>512/384</td>
<td>512</td>
<td>1024</td>
<td>128</td>
<td>64</td>
<td>No</td>
</tr>
<tr>
<td>Tiger(2)</td>
<td>192/160/128</td>
<td>192</td>
<td>512</td>
<td>64</td>
<td>64</td>
<td>No</td>
</tr>
<tr>
<td>WHIRLPOOL</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>256</td>
<td>8</td>
<td>No</td>
</tr>
</tbody>
</table>
Outline

Classical Symmetric Encryptions
  DES
  3-DES
  AES
  IDEA

Modes
  ECB
  CBC
  CFB
  OFB

Asymmetric vs Symmetric
Diffie-Hellman

Hash Functions

Applications

Conclusion
Utility of Cryptography in Real life

- Hash function, e.g. Software Installation
- Asymmetric Encryption for establishing a Session Key
- Symmetric Encryption for GSM communication
- Signature for Authentication, e.g. CB
Hash function, e.g. Software Installation

1. Download on server 1 the software.
2. Download on server 2 the hash of the software.
3. Check the integrity of the software.

Integrity of the downloaded file.
Asymmetric Encryption for establishing a Session Key

1. Server has a public and private key
2. Computer asks for a secure connection
3. Server sends him his public key
4. Client chooses a symmetric key which is sent encrypted by the public key of the server
Symmetric Encryption for GSM communication

1. Message is encrypted and sent by Alice.
2. The antenna receives the message then uncrypted.
3. Message is encrypted by the antenna with the second key.
4. Second mobile uncrypted the communication.

SIM card contains a shared secret key used for authenticating phones and operators, then creating key session for communication.
Signature for Authentication of Credit Card

Off-line authentication of the card.

1. Credit Card has informations I and S a signature of H(I).
2. Machine reads I and S.
3. Machine checks if h(I) = Unsign(S).

Example: SHA1 ...
Outline

Classical Symetric Encryptions
  DES
  3-DES
  AES
  IDEA

Modes
  ECB
  CBC
  CFB
  OFB

Asymmetric vs Symetric
Diffie-Hellman
Hash Functions
Applications

Conclusion
Summary

Today

- Classical Symmetric Encryption
- Encryption Modes
- Comparison between Symmetric and Asymmetric encryption
- Diffie Hellman
- Hash functions
- Applications
Next Time

- Security notions
Thank you for your attention

Questions?