Symmetric Encryption et al.

Pascal Lafourcade

*Université Joseph Fourier, Verimag*

4th November 2008
Last Time

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

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

2 Modes
   - ECB
   - CBC
   - CFB
   - OFB

3 Asymmetric vs Symmetric

4 Diffie-Hellman

5 Hash Functions

6 Applications

7 Conclusion
Outline

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

2. Modes
   - ECB
   - CBC
   - CFB
   - OFB

3. Asymmetric vs Symmetric

4. Diffie-Hellman

5. Hash Functions

6. Applications

7. Conclusion
Data Encryption Standard, 1993

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

- First cryptographic standard.
  - 1977 US federal standard (US Bureau of Standards)
  - 1981 ANSI private sector standard
Symmetric Encryption et al.
Classical Symmetric Encryptions

DES

**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.
Symmetric Encryption et al.
Classical Symmetric Encryptions

DES

**DES — 1 round**

(L_{i-1}, R_{i-1}) → Expansion Permutation → S-Box Substitution → P-Box Permutation

K_{i-1} → Left Shift → Compression Permutation

(L_{i}, R_{i}) → K_{i}

(b_{1}b_{6}, b_{2}b_{3}b_{4}b_{5}), C_j represents the binary value in the row b_{1}b_{6} and column b_{2}b_{3}b_{4}b_{5} of the S_j 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

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

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

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

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

DES exhibits the complementation property, namely that

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

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 4 weak keys: $E_k(E_k(x)) = x$.
  - all zeros (0x0000000000000000)
  - all ones (0xFFFFFFFFFFFFFFFF)
  - '0xE1E1E1E1F0F0F0F0'
  - '0x1E1E1E1E0F0F0F0F'

- Existence of 6 pairs of semi-weak keys: $E_{k_1}(E_{k_2}(x)) = x$.
  - 0x011F011F010E010E and 0x1F011F010E010E01
  - 0x01E001E001F101F1 and 0xE001E001F101F101
  - 0x01FE01FE01FE01FE and 0xFE01FE01FE01FE01
  - 0x1FE01FE00EF10EF1 and 0xE01FE01FF10EF10E
  - 0x1FFE1FFE0EFE0EFE and 0xFE1FFE1FFE0EFE0E
  - 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
  ⇒ brute-force search with $2^{112}$ operations.
Advanced Encryption Standard

- Blocksize = 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
AES: ShiftRows
AES: MixColumns
AES: AddRoundKey
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.

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.
Notation

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

IDEA
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

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

2. Modes
   - ECB
   - CBC
   - CFB
   - OFB

3. Asymmetric vs Symmetric

4. Diffie-Hellman

5. Hash Functions

6. Applications

7. 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

**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
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
Symmetric Encryption et al.

Modes

CFB

Cipher Feedback (CFB) mode decryption
Output feedback (OFB)

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

\[ C_i = P_i \oplus O_i \]

\[ P_i = C_i \oplus O_i \]

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

\[ O_0 = IV \]
Output Feedback (OFB) mode encryption
Output Feedback (OFB) mode decryption
ECB vs Others
Outline

1 Classical Symmetric Encryptions
   DES
   3-DES
   AES
   IDEA

2 Modes
   ECB
   CBC
   CFB
   OFB

3 Asymmetric vs Symetric

4 Diffie-Hellman

5 Hash Functions

6 Applications

7 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>DVD 47 G.B</th>
<th>Blu-Ray 25 GB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>encrypt</td>
<td>decrypt</td>
</tr>
<tr>
<td>RSA 2048(1)</td>
<td>22 min</td>
<td>24 h</td>
</tr>
<tr>
<td>RSA 1024(1)</td>
<td>21 min</td>
<td>10 h</td>
</tr>
<tr>
<td>AES CTR(2)</td>
<td>20 sec</td>
<td>20 sec</td>
</tr>
</tbody>
</table>
Outline

1. Classical Symmetric Encryptions
   - DES
   - 3-DES
   - AES
   - IDEA
2. Modes
   - ECB
   - CBC
   - CFB
   - OFB
3. Asymetric vs Symetric
4. Diffie-Hellman
5. Hash Functions
6. Applications
7. Conclusion
The Diffie-Hellman Key Exchange Protocol

Let $g$ be a generator of a cyclic group of prime order $q$.

\[ A \rightarrow B : g^a \]
\[ B \rightarrow A : g^b \]
\[ A \rightarrow B : \{N\}_{g^{ab}} \]
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 Discret 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 $A$,

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

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 $\mathcal{A}$,

$$\text{Adv}^{\text{CDH}}(\mathcal{A}) = \Pr \left[ \mathcal{A}(g^x, g^y) \to g^{xy} \middle| x, y \stackrel{R}{\leftarrow} [1, q] \right]$$

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 $A$, the advantage of $A$

$$\text{Adv}_{DDH}^A = Pr\left[A(g^x, g^y, g^{xy}) \rightarrow 1 \mid x, y \xleftarrow{R} [1, q]\right] - Pr\left[A(g^x, g^y, g^r) \rightarrow 1 \mid x, y, r \xleftarrow{R} [1, q]\right]$$

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**

<table>
<thead>
<tr>
<th>Prop</th>
<th>( \text{Solve } DL \Rightarrow \text{Solve } CDH \Rightarrow \text{Solve } DDH ). (Exercice)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prop (Moaurer &amp; Wolf)</td>
<td>( \text{For many groups, } DL \Leftrightarrow CDH )</td>
</tr>
<tr>
<td>Prop (Joux &amp; Wolf)</td>
<td>( \text{There are groups for which } DDH \text { is easier than } CDH. )</td>
</tr>
</tbody>
</table>
Proofs by Reduction

**Solve DL ⇒ 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 ⇒ 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 cryptosystems [ElGamal, Cramer & Shoup]
- Pseudo-random functions [Noar & Reingold, Canetti]
- Pseudo-random generators [Blum & Micali]
- (Group) key exchange protocols [many]
Outline

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

2. Modes
   - ECB
   - CBC
   - CFB
   - OFB

3. Asymmetric vs Symmetric

4. Diffie-Hellman

5. Hash Functions

6. Applications

7. Conclusion
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.

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

$\mathcal{H}(Alice)$ =

**Definition (Preimage resistance (One-way) OWHF)**

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

$$h(x) = y$$
“Classifications” of Hash Functions

Unkeyed Hash function

- Modification Code Detection (MDC)
- Datat integrity
- Fingerprints of messages
- Other applications

Keyed Hash function

- Message Authentication Code (MAC)
- Password Verification in unencrypted password-image files.
- Key confirmation or establishment
- Timestamping
- Others applications
Properties of hash functions

2nd preimage 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 preimage resistance**

If \( h \) is not 2nd preimage 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 preimage resistance. Example let \( g \) be collision resistant we build \( h \) such that

\[
    h(x) = \begin{cases} 
    1 || x & \text{if } x \text{ has bitlenght } n \\
    0 || g(x) & \text{otherwise} 
    \end{cases}
\]

\( h \) is collision resistance but not preimage 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 $x = x_1 x_2 \cdots x_l$
- Iterated processing
  - Compression function $f$
    - $f(H_{i-1}, x_i) = H_i$
    - $H_0 = IV$
- Output $h(x) = g(H_l)$
Basic construction of hash functions (Merkle-Damgrd)

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

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 bitlength 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

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

2 Modes
   - ECB
   - CBC
   - CFB
   - OFB

3 Asymetric vs Symetric

4 Diffie-Hellman

5 Hash Functions

6 Applications

7 Conclusion
Utility of Cryptography in Real life

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

SIM card contains a shared secret key used for authenticating phones and operators, then creating key session for 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.
Hash function, e.g. Software Installation

Integrity of the downloaded file.

1. Download on server 1 the software.
2. Download on server 2 the hash of the software.
3. Check the integrity of the software.
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 \( h(I) = \text{Unsign}(S) \).

Example: SHA1...
Outline

1. Classical Symmetric Encryptions
   - DES
   - 3-DES
   - AES
   - IDEA
2. Modes
   - ECB
   - CBC
   - CFB
   - OFB
3. Asymmetric vs Symmetric
4. Diffie-Hellman
5. Hash Functions
6. Applications
7. Conclusion
Summary

Today

- Classical Symmetric Encryption
- Encryption Modes
- Comparison between Symmetric and Asymmetric encryption
- Applications
Next Time

- Security notions
Thank you for your attention

Questions?