Block ciphers

What is a block cipher?
Block ciphers: crypto work horse

 Canonical examples:
1. 3DES: $n = 64$ bits, $k = 168$ bits
2. AES: $n = 128$ bits, $k = 128, 192, 256$ bits
Block Ciphers Built by Iteration

$R(k_1, \cdot) \rightarrow R(k_2, \cdot) \rightarrow R(k_3, \cdot) \rightarrow R(k_n, \cdot)$

$m \rightarrow C$

$R(k, m)$ is called a round function

for 3DES (n=48), for AES-128 (n=10)
# Performance:

AMD Opteron, 2.2 GHz (Linux)

<table>
<thead>
<tr>
<th>Cipher</th>
<th>Block/key size</th>
<th>Speed (MB/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC4</td>
<td></td>
<td>126</td>
</tr>
<tr>
<td>Salsa20/12</td>
<td></td>
<td>643</td>
</tr>
<tr>
<td>Sosemanuk</td>
<td></td>
<td>727</td>
</tr>
<tr>
<td>3DES</td>
<td>64/168</td>
<td>13</td>
</tr>
<tr>
<td>AES-128</td>
<td>128/128</td>
<td>109</td>
</tr>
</tbody>
</table>
Abstractly: PRPs and PRFs

• Pseudo Random Function (PRF) defined over (K,X,Y):
  \[ F: K \times X \to Y \]
  such that there exists “efficient” algorithm to evaluate \( F(k,x) \)

• Pseudo Random Permutation (PRP) defined over (K,X):
  \[ E: K \times X \to X \]
  such that:
  1. There exists “efficient” deterministic algorithm to evaluate \( E(k,x) \)
  2. The function \( E(k, \cdot) \) is one-to-one
  3. There exists “efficient” inversion algorithm \( D(k,y) \)
Running example

• **Example PRPs:** 3DES, AES, ...

  AES: \( K \times X \rightarrow X \) where \( K = X = \{0,1\}^{128} \)

  3DES: \( K \times X \rightarrow X \) where \( X = \{0,1\}^{64} \), \( K = \{0,1\}^{168} \)

• Functionally, any PRP is also a PRF.
  – A PRP is a PRF where \( X=Y \) and is efficiently invertible.
Secure PRFs

- Let $F: K \times X \rightarrow Y$ be a PRF

$$Funs[X,Y]: \text{the set of all functions from } X \text{ to } Y$$

$$S_F = \{ F(k,\cdot) \text{ s.t. } k \in K \} \subseteq Funs[X,Y]$$

- Intuition: a PRF is secure if a random function in $Funs[X,Y]$ is indistinguishable from a random function in $S_F$
Secure PRFs

- Let $F: K \times X \rightarrow Y$ be a PRF

$$F\text{uns}[X,Y]: \text{ the set of all functions from } X \text{ to } Y$$

$$S_F = \{ F(k,\cdot) \text{ s.t. } k \in K \} \subseteq F\text{uns}[X,Y]$$

- **Intuition:** a PRF is **secure** if

a random function in $F\text{uns}[X,Y]$ is indistinguishable from a random function in $S_F$
Secure PRPs  (secure block cipher)

• Let $E: K \times X \rightarrow Y$ be a PRP

\[
\begin{array}{l}
\text{Perms}[X]: \quad \text{the set of all one-to-one functions from } X \text{ to } Y \\
S_F = \{ E(k,\cdot) \text{ s.t. } k \in K \} \subseteq \text{Perms}[X,Y]
\end{array}
\]

• Intuition: a PRP is secure if a random function in Perms[X] is indistinguishable from a random function in $S_F$
Let $F: K \times X \rightarrow \{0,1\}^{128}$ be a secure PRF.

Is the following $G$ a secure PRF?

$$G(k, x) = \begin{cases} 
0^{128} & \text{if } x=0 \\
F(k,x) & \text{otherwise} 
\end{cases}$$

- No, it is easy to distinguish $G$ from a random function
- Yes, an attack on $G$ would also break $F$
- It depends on $F$
An easy application: PRF $\Rightarrow$ PRG

Let $F: K \times \{0,1\}^n \rightarrow \{0,1\}^n$ be a secure PRF.

Then the following $G: K \rightarrow \{0,1\}^{nt}$ is a secure PRG:

$$G(k) = F(k,0) \parallel F(k,1) \parallel \cdots \parallel F(k,t-1)$$

Key property: parallelizable

Security from PRF property: $F(k, \cdot)$ indist. from random function $f(\cdot)$
End of Segment
Block ciphers

The data encryption standard (DES)
Block ciphers: crypto work horse

Canonical examples:

1. 3DES: \( n = 64 \) bits, \( k = 168 \) bits
2. AES: \( n = 128 \) bits, \( k = 128, 192, 256 \) bits
R(k, m) is called a round function

for 3DES (n=48), for AES-128 (n=10)
The Data Encryption Standard (DES)

• Early 1970s: Horst Feistel designs Lucifer at IBM
  key-len = 128 bits ; block-len = 128 bits

• 1973: NBS asks for block cipher proposals.
  IBM submits variant of Lucifer.

• 1976: NBS adopts DES as a federal standard
  key-len = 56 bits ; block-len = 64 bits

• 1997: DES broken by exhaustive search

• 2000: NIST adopts Rijndael as AES to replace DES

Widely deployed in banking (ACH) and commerce
DES: core idea – Feistel Network

Given functions  \( f_1, \ldots, f_d: \{0,1\}^n \rightarrow \{0,1\}^n \)

Goal: build invertible function  \( F: \{0,1\}^{2n} \rightarrow \{0,1\}^{2n} \)

In symbols:
\[
\begin{align*}
R_i &= f_i(R_{i-1}) \oplus L_{i-1} \\
L_i &= R_{i-1}
\end{align*}
\]
Claim: for all $f_1, \ldots, f_d$: $\{0,1\}^n \rightarrow \{0,1\}^n$

Feistel network $F: \{0,1\}^{2n} \rightarrow \{0,1\}^{2n}$ is invertible

Proof: construct inverse

$$R_{i-1} = L_i$$

$$L_{i-1} =$$
Claim: for all $f_1, \ldots, f_d: \{0,1\}^n \rightarrow \{0,1\}^n$

Feistel network $F: \{0,1\}^{2n} \rightarrow \{0,1\}^{2n}$ is invertible

Proof: construct inverse
Decryption circuit

- Inversion is basically the same circuit, with $f_1, \ldots, f_d$ applied in reverse order.

- General method for building invertible functions (block ciphers) from arbitrary functions.

- Used in many block ciphers ... but not AES
“Thm:” (Luby-Rackoff ‘85):

\[ f: K \times \{0,1\}^n \rightarrow \{0,1\}^n \text{ a secure PRF} \]

\[ \Rightarrow \text{ 3-round Feistel } F: K^3 \times \{0,1\}^{2n} \rightarrow \{0,1\}^{2n} \text{ a secure PRP} \]
DES: 16 round Feistel network

\[ f_1, \ldots, f_{16}: \{0,1\}^{32} \rightarrow \{0,1\}^{32}, \quad f_i(x) = F(k_i, x) \]

To invert, use keys in reverse order.
The function $F(k_i, x)$

S-box: function $\{0,1\}^6 \rightarrow \{0,1\}^4$, implemented as look-up table.
The S-boxes

\[ S_i: \{0,1\}^6 \rightarrow \{0,1\}^4 \]

<table>
<thead>
<tr>
<th>Inner bits</th>
<th>Middle 4 bits of input</th>
<th>Outer bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0010 1100 0100 0001 0111 1010 1011 0110 1000 0101 0011 1111 1101 1101 0000 1110 1001</td>
<td></td>
</tr>
<tr>
<td>0011</td>
<td>1110 1011 0010 1100 0100 0111 1101 0001 0101 0000 1111 1010 0011 1001 1000 0110</td>
<td></td>
</tr>
<tr>
<td>0100</td>
<td>0100 0010 0001 1011 1010 1101 0111 1000 1111 1001 1100 0101 0110 0011 0000 1110</td>
<td></td>
</tr>
<tr>
<td>0111</td>
<td>1011 1000 1100 0111 0001 1110 0010 1101 0110 1111 0000 1001 1010 0100 0101 0011</td>
<td></td>
</tr>
</tbody>
</table>
Example: a bad S-box choice

Suppose:

\[ S_i(x_1, x_2, ..., x_6) = (x_2 \oplus x_3, x_1 \oplus x_4 \oplus x_5, x_1 \oplus x_6, x_2 \oplus x_3 \oplus x_6) \]

or written equivalently: \[ S_i(x) = A_i \cdot x \pmod{2} \]

We say that \( S_i \) is a linear function.
Example: a bad S-box choice

Then entire DES cipher would be linear:  \( \exists \) fixed binary matrix \( B \) s.t.

\[
\text{DES}(k,m) = 64 \begin{bmatrix} m \end{bmatrix} B \begin{bmatrix} k_1 \ k_2 \ \cdots \ k_{16} \end{bmatrix} = c \quad \text{(mod 2)}
\]

But then:  \( \text{DES}(k,m_1) \oplus \text{DES}(k,m_2) \oplus \text{DES}(k,m_3) = \text{DES}(k, m_1 \oplus m_2 \oplus m_3) \)
Choosing the S-boxes and P-box

Choosing the S-boxes and P-box at random would result in an insecure block cipher (key recovery after $\approx 2^{24}$ outputs) [BS’89]

Several rules used in choice of S and P boxes:
• No output bit should be close to a linear func. of the input bits
• S-boxes are 4-to-1 maps
  •
  •
End of Segment
Block ciphers

Exhaustive Search Attacks
Exhaustive Search for block cipher key

Goal: given a few input output pairs \((m_i, c_i = E(k, m_i))\) \(i=1,..,3\) find key \(k\).

Lemma: Suppose DES is an ideal cipher

\((2^{56} \text{ random invertible functions})\)

Then \(\forall \ m, c \ \text{there is at most one key } k \text{ s.t. } c = \text{DES}(k, m)\)

Proof: with prob. \(\geq 1 - 1/256 \approx 99.5\%\)

\[
P_k[\exists k' \neq k : c = \text{DES}(k, m) = \text{DES}(k', m)] \leq \sum_{k' \in \{0,1\}^{56}} P_k[\text{DES}(k, m) = \text{DES}(k', m)] \leq 2^{56} \cdot \frac{1}{2^{56}} = \frac{1}{2^8}
\]
Exhaustive Search for block cipher key

For two DES pairs \((m_1, c_1=DES(k, m_1))\), \((m_2, c_2=DES(k, m_2))\)
unicity prob. \(\approx 1 - 1/2^{71}\)

For AES-128: given two inp/out pairs, unicity prob. \(\approx 1 - 1/2^{128}\)

\[\Rightarrow\] two input/output pairs are enough for exhaustive key search.
### DES challenge

**msg = “The unknown messages is: XXXX ... “**

<table>
<thead>
<tr>
<th>CT</th>
<th>c₁</th>
<th>c₂</th>
<th>c₃</th>
<th>c₄</th>
</tr>
</thead>
</table>

**Goal:** find $k \in \{0, 1\}^{56}$ s.t. $\text{DES}(k, m_i) = c_i$ for $i=1,2,3$

1997: Internet search -- **3 months**
1998: EFF machine (deep crack) -- **3 days** (250K $)
1999: combined search -- **22 hours**
2006: COPACOBANA (120 FPGAs) -- **7 days** (10K $)

$\Rightarrow$ 56-bit ciphers should not be used !! (128-bit key $\Rightarrow 2^{72}$ days)
Strengthening DES against ex. search

Method 1: **Triple-DES**

- Let \( E : K \times M \rightarrow M \) be a block cipher

- Define \( 3E : K^3 \times M \rightarrow M \) as

\[
3E((k_1,k_2,k_3), m) = E(k_1, D(k_2, E(k_3, m)))
\]

\( k_1 = k_2 = k_3 \Rightarrow \text{single DES} \)

For 3DES: key-size = 3\times56 = 168 bits. \hspace{1cm} 3\times\text{slower than DES.}

(simple attack in time \( \approx 2^{118} \))
Why not double DES?

- Define \( 2E((k_1, k_2), m) = E(k_1, E(k_2, m)) \)

**Attack:** \( M = (m_1, ..., m_{10}) \), \( C = (c_1, ..., c_{10}) \).

- Step 1: build table.
  - Sort on 2\(^{nd}\) column

\[
\begin{align*}
\text{Find } (k_1, k_2) \text{ s.t. } & E(k_1, E(k_2, M)) = C \\
& \text{Equivalently: } E(k_2, M) = D(k_1, C)
\end{align*}
\]

- \( k^0 = 00...00 \)
- \( k^1 = 00...01 \)
- \( k^2 = 00...10 \)
- \( \vdots \)
- \( k^N = 11...11 \)

\( 2^{56} \) entries
Meet in the middle attack

Attack: \( M = (m_1, ..., m_{10}) \), \( C = (c_1, ..., c_{10}) \)

- step 1: build table.

- Step 2: for all \( k \in \{0,1\}^{56} \) do:

  test if \( D(k, C) \) is in 2\(^{nd} \) column.

  if so then \( E(k^i, M) = D(k, C) \) \( \Rightarrow (k^i, k) = (k_2, k_1) \)
Meet in the middle attack

Time = $2^{56}\log(2^{56}) + 2^{56}\log(2^{56}) < 2^{63} \ll 2^{112}$, space $\approx 2^{56}$

Same attack on 3DES: Time = $2^{118}$, space $\approx 2^{56}$
Method 2: DESX

E : K × \{0,1\}^n \rightarrow \{0,1\}^n a block cipher

Define EX as

\[
EX((k_1,k_2,k_3), m) = k_1 \oplus E(k_2, m \oplus k_3)
\]

For DESX: key-len = 64+56+64 = 184 bits

... but easy attack in time \(2^{64+56} = 2^{120}\) (homework)

Note: \(k_1 \oplus E(k_2, m)\) and \(E(k_2, m \oplus k_1)\) does nothing !!
End of Segment
Block ciphers

More attacks on block ciphers
Attacks on the implementation

1. Side channel attacks:
   - Measure **time** to do enc/dec, measure **power** for enc/dec

2. Fault attacks:
   - Computing errors in the last round expose the secret key $k$

$\Rightarrow$ do not even implement crypto primitives yourself ...

[Kocher, Jaffe, Jun, 1998]
Linear and differential attacks [BS’89,M’93]

Given many input/output pairs, can recover key in time less than $2^{56}$.

**Linear cryptanalysis** (overview): let $c = \text{DES}(k, m)$

Suppose for random $k, m$:

$$\Pr\left[ \bigoplus_{i=1}^{r} m[i] \oplus \bigoplus_{j=1}^{v} c[j] = \bigoplus_{l=1}^{u} k[l] \right] = \frac{1}{2} + \varepsilon$$

For some $\varepsilon$. For DES, this exists with $\varepsilon = 1/2^{21} \approx 0.0000000477$
Linear attacks

$$\Pr \left[ m[i_1] \oplus \cdots \oplus m[i_r] \oplus c[j_1] \oplus \cdots \oplus c[j_v] = k[l_1] \oplus \cdots \oplus k[l_u] \right] = \frac{1}{2} + \epsilon$$

Thm: given $1/\epsilon^2$ random $(m, c=DES(k, m))$ pairs then

$$k[l_1, \ldots, l_u] = \text{MAJ} \left[ m[i_1, \ldots, i_r] \oplus c[j_1, \ldots, j_v] \right]$$

with prob. $\geq 97.7\%$

$\Rightarrow$ with $1/\epsilon^2$ inp/out pairs can find $k[l_1, \ldots, l_u]$ in time $\approx 1/\epsilon^2$. 

Dan Boneh
Linear attacks

For DES, \( \varepsilon = 1/2^{21} \Rightarrow \)

with \( 2^{42} \) inp/out pairs can find \( k[l_1, ..., l_u] \) in time \( 2^{42} \)

Roughly speaking: can find 14 key “bits” this way in time \( 2^{42} \)

Brute force remaining \( 56-14=42 \) bits in time \( 2^{42} \)

Total attack time \( \approx 2^{43} (<< 2^{56}) \) with \( 2^{42} \) random inp/out pairs
Lesson

A tiny bit of linearly in $S_5$ lead to a $2^{42}$ time attack.

⇒ don’t design ciphers yourself !!
Quantum attacks

Generic search problem:

Let $f : X \rightarrow \{0,1\}$ be a function.

Goal: find $x \in X$ s.t. $f(x)=1$.

Classical computer: best generic algorithm time $= O(|X|)$

Quantum computer [Grover ’96]: time $= O(|X|^{1/2})$

Can quantum computers be built: unknown
Quantum exhaustive search

Given $m, c = E(k, m)$ define

$$f(k) = \begin{cases} 
1 & \text{if } E(k, m) = c \\
0 & \text{otherwise}
\end{cases}$$

Grover $\Rightarrow$ quantum computer can find $k$ in time $O( |K|^{1/2} )$

DES: time $\approx 2^{28}$, AES-128: time $\approx 2^{64}$

quantum computer $\Rightarrow$ 256-bits key ciphers (e.g. AES-256)
End of Segment
Block ciphers

The AES block cipher
The AES process

• 1997: NIST publishes request for proposal


• 1999: NIST chooses 5 finalists

• 2000: NIST chooses Rijndael as AES (designed in Belgium)

Key sizes: 128, 192, 256 bits. Block size: 128 bits
AES is a Subs-Perm network (not Feistel)

\[
\text{input} \oplus \begin{array}{c}
S_1 \\
S_2 \\
S_3 \\
\vdots \\
S_8
\end{array} \text{ subs. layer} \rightarrow \\
\begin{array}{c}
k_1 \\
S_1 \\
S_2 \\
S_3 \\
S_8
\end{array} \rightarrow \\
\begin{array}{c}
k_2 \\
S_1 \\
S_2 \\
S_3 \\
S_8
\end{array} \rightarrow \\
\begin{array}{c}
\vdots \\
\vdots \\
\vdots \\
\vdots
\end{array} \rightarrow \\
\begin{array}{c}
k_n \\
S_1 \\
S_2 \\
S_3 \\
S_8
\end{array} \text{ perm. layer} \rightarrow \\
\oplus \text{ output}
\]
AES-128 schematic

key expansion: 16 bytes $\rightarrow$ 176 bytes

input

key

16 bytes
The round function

- **ByteSub:** a 1 byte S-box. 256 byte table (easily computable)

- **ShiftRows:**

- **MixColumns:**
## Code size/performance tradeoff

<table>
<thead>
<tr>
<th></th>
<th>Code size</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-compute round functions (24KB or 4KB)</td>
<td>largest</td>
<td>fastest: table lookups and xors</td>
</tr>
<tr>
<td>Pre-compute S-box only (256 bytes)</td>
<td>smaller</td>
<td>slower</td>
</tr>
<tr>
<td>No pre-computation</td>
<td>smallest</td>
<td>slowest</td>
</tr>
</tbody>
</table>

Dan Boneh
Example: Javascript AES

AES in the browser:

AES library (6.4KB)
no pre-computed tables

Prior to encryption:
pre-compute tables

Then encrypt using tables

http://crypto.stanford.edu/sjcl/
AES in hardware

AES instructions in Intel Westmere:

• `aesenc, aesenclast`: do one round of AES
  128-bit registers: `xmm1=state, xmm2=round key`
  `aesenc xmm1, xmm2`; puts result in `xmm1`

• `aeskeygenassist`: performs AES key expansion

• Claim 14 x speed-up over OpenSSL on same hardware

Similar instructions on AMD Bulldozer
Attacks

Best key recovery attack:

\[ \text{four times better than ex. search \ [BKR'11]} \]

Related key attack on AES-256: \ [BK'09] \n
Given \( 2^{99} \) inp/out pairs from \textbf{four related keys} in AES-256

\( \text{can recover keys in time } \approx 2^{99} \)
End of Segment
Block ciphers from PRGs
Can we build a PRF from a PRG?

Let $G: K \rightarrow K^2$ be a secure PRG

Define 1-bit PRF $F: K \times \{0,1\} \rightarrow K$ as

$$F(k, x \in \{0,1\}) = G(k)[x]$$

Thm: If $G$ is a secure PRG then $F$ is a secure PRF

Can we build a PRF with a larger domain?
Extending a PRG

Let $G: K \rightarrow K^2$. Define $G_1: K \rightarrow K^4$ as $G_1(k) = G(G(k)[0]) || G(G(k)[1])$

We get a 2-bit PRF:

$$F(k, x \in \{0,1\}^2) = G_1(k)[x]$$
$G_1$ is a secure PRG

$G_1(k)$

random in $K^4$

$G(k)[0]$  $G(k)[1]$

$00$  $01$  $10$  $11$

$G_1$ is a secure PRG

$G_1(k)$

random in $K^4$
Extending more

Let \( G : K \rightarrow K^2 \).

Define \( G_2 : K \rightarrow K^8 \) as \( G_2(k) = G(k)[0] \cdot G(k)[1] \).

We get a 3-bit PRF

\[
\begin{array}{cccccc}
000 & 001 & 010 & 011 & 100 & 101 \\
110 & 111 & & & & \\
\end{array}
\]

Eval \( F(k, 101) \) as follows:
Extending even more: the GGM PRF

Let $G: K \rightarrow K^2$. Define PRF $F: K \times \{0,1\}^n \rightarrow K$ as

For input $x = x_0 \ x_1 \ldots \ x_{n-1} \in \{0,1\}^n$ do:

Security: $G$ a secure PRG $\Rightarrow$ $F$ is a secure PRF on $\{0,1\}^n$.

Not used in practice due to slow performance.
Secure block cipher from a PRG?

Can we build a secure PRP from a secure PRG?

- No, it cannot be done
- Yes, just plug the GGM PRF into the Luby-Rackoff theorem
- It depends on the underlying PRG
End of Segment