Cryptography: Block Ciphers

Edward J. Schwartz
Carnegie Mellon University

Credits:
Slides originally designed by David Brumley. Many other slides are from Dan Boneh’s June 2012 Coursera crypto class.
What is a block cipher?

Block ciphers are the crypto work horse

Canonical examples:
1. 3DES: \( n = 64 \text{ bits}, \quad k = 168 \text{ bits} \)
2. AES: \( n = 128 \text{ bits}, \quad k = 128, 192, 256 \text{ bits} \)
Block ciphers built by iteration

key k

key expansion

key k_1

key k_2

key k_3

key k_n

m \rightarrow R(k_1, \cdot) \rightarrow m_1 \rightarrow R(k_2, \cdot) \rightarrow m_2 \rightarrow R(k_3, \cdot) \rightarrow m_3 \rightarrow R(k_n, \cdot) \rightarrow c

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

Ex: 3DES (n=48), AES128 (n=10)
### Performance: Stream vs. block ciphers

**Crypto++ 5.6.0 [Wei Dai]**

**AMD Opteron, 2.2 GHz (Linux)**

<table>
<thead>
<tr>
<th>Cipher</th>
<th>Block/key size</th>
<th>Throughput [MB/s]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Stream</strong></td>
<td></td>
<td></td>
</tr>
<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><strong>Block</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3DES</td>
<td>64/168</td>
<td>13</td>
</tr>
<tr>
<td>AES128</td>
<td>128/128</td>
<td>109</td>
</tr>
</tbody>
</table>
Block ciphers

The Data Encryption Standard (DES)
History of DES

• **1970s:** Horst Feistel designs Lucifer at IBM
  key = 128 bits, block = 128 bits

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

• **1976:** NBS adopts DES as federal standard
  key = 56 bits, block = 64 bits

• **1997:** DES broken by exhaustive search

• **2000:** NIST adopts Rijndael as AES to replace DES. AES currently widely deployed in banking, commerce and Web
DES: core idea – Feistel network

Given one-way 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:

$$R_i = f_i(R_{i-1}) \oplus L_{i-1}$$
$$L_i = R_{i-1}$$
Feistel network - inverse

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

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

Proof: construct inverse

\[
\begin{align*}
R_i &= R_{i+1} \\
L_i &= L_{i+1}
\end{align*}
\]
Decryption circuit

- Inversion is basically the same circuit, with $f_1$, ..., $f_d$ applied in reverse order.
- General method for building invertible functions (block ciphers) from arbitrary functions.
- Used in many block ciphers ... but not AES.
Recall from Last Time:

Block Ciphers are (Modeled As) PRPs

Pseudo Random Permutation (PRP) defined over (K,X)

\[ E : K \times X \rightarrow X \]

such that:

1. Exists “efficient” deterministic algorithm to evaluate \( E(k,x) \)
2. The function \( E(k, \cdot) \) is one-to-one
3. Exists “efficient” inversion algorithm \( D(k,y) \)
Luby-Rackoff Theorem (1985)

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

\[ \Rightarrow 3\text{-round Feistel} \quad F : K^3 \times \{0, 1\}^{2n} \rightarrow \{0, 1\}^{2n} \]

is a secure PRP
DES: 16 round Feistel network

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

16 round Feistel network

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

The function $F(k_i, x)$ takes a 32-bit input $x$ and a 48-bit key $k_i$. It computes $x'$ and then applies the S-boxes to the result. The S-boxes are functions that map 6-bit inputs to 4-bit outputs. The S-boxes are implemented as a lookup table.

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

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

e.g., 011011 \rightarrow 1001
The S-boxes

• Alan Konheim (one of the designers of DES) commented, "We sent the S-boxes off to Washington. They came back and were all different."

• 1990: (Re-)Discovery of differential cryptanalysis
  – DES S-boxes resistant to differential cryptanalysis!
  – Both IBM and NSA knew of attacks, but they were classified
Block cipher attacks
Exhaustive Search for block cipher key

**Goal:** given a few input output pairs

\[(m_i, c_i = E(k, m_i)) \text{ i=1,...,n} \] find key k.

**Attack:** Brute force to find the key k.

**Homework:** What is the probability that the key k found with one \(<m,c>\) pair is correct? For two pairs?
DES challenge

\[ \text{msg} = \text{“The unknown messages is: XXXXXXXXX...“} \]

\[ \text{CT} = \begin{array}{cccc}
  c_1 & c_2 & c_3 & c_4 \\
\end{array} \]

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

Proof: Reveal \( \text{DES}^{-1}(k, c_4) \)

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
<th>Time</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1976</td>
<td>DES adopted as federal standard</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1997</td>
<td>Distributed search</td>
<td>3 months</td>
<td></td>
</tr>
<tr>
<td>1998</td>
<td>EFF deep crack</td>
<td>3 days</td>
<td>$250,000</td>
</tr>
<tr>
<td>1999</td>
<td>Distributed search</td>
<td>22 hours</td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td>COPACOBANA (120 FPGAs)</td>
<td>7 days</td>
<td>$10,000</td>
</tr>
</tbody>
</table>

\( \Rightarrow \) 56-bit ciphers should not be used \((128\text{-bit key} \Rightarrow 2^{72} \text{ days})\)
Strengthening DES

Method 1: **Triple-DES**

Let $E : K \times M \to M$ be a block cipher

Define $3E : K^3 \times M \to M$ as:

$$3E((k_1, k_2, k_3), m) = E(k_1, D(k_2, E(k_3, m)))$$

**3DES**
- Key-size: $3 \times 56 = 168$ bits
- $3 \times$ slower than DES
- Simple attack in time: $\approx 2^{118}$

$k_1 = k_2 = k_3 \Rightarrow$ DES
Why not 2DES?

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

key-len = 112 bits for 2DES

Naïve Attack: \( M = (m_1, ..., m_{10}), C = (c_1, ..., c_{10}) \).
For each \( k_2 \in \{0,1\}^{56} \):
  For each \( k_1 \in \{0,1\}^{56} \):
    if \( E(k_2, E(k_1, m_i)) = c_i \) then \( (k_2, k_1) \)

\[ 2^{112} \text{ checks} \]
\[ c'' = c? \]
Meet in the middle attack

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

key-len = 112 bits for 2DES

Idea: key found when $c' = c'': E(k_i, m) = D(k_j, c)$
Meet in the middle attack

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

key-len = 112 bits for 2DES

\[
\begin{array}{c}
\text{m} \rightarrow E(k_2, \cdot) \rightarrow E(k_1, \cdot) \rightarrow c \\
\end{array}
\]

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

• step 1: build table.
  sort on 2\textsuperscript{nd} column
  maps \( c' \) to \( k_2 \)

\[
\begin{array}{c|c|c}
  k^0 \rightarrow 00\ldots00 & E(k^0 , M) \\
  k^1 \rightarrow 00\ldots01 & E(k^1 , M) \\
  k^2 \rightarrow 00\ldots10 & E(k^2 , M) \\
  \vdots & \vdots \\
  k^N \rightarrow 11\ldots11 & E(k^N , M) \\
\end{array}
\]

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

\[ M = (m_1, ..., m_{10}) \quad \text{and} \quad C = (c_1, ..., c_{10}) \]

- **Step 1:** Build table.

- **Step 2:** For each \( k \in \{0,1\}^{56} \):
  
  Test if \( D(k, c) \) is in the 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

\[ \text{Time} = 2^{56} \log(2^{56}) + 2^{56} \log(2^{56}) < 2^{63} \ll 2^{112} \]

[Build & Sort Table] [Search Entries]

Space \approx 2^{56} [Table Size]

Same attack on 3DES: \quad \text{Time} = 2^{118}, \ \text{Space} \approx 2^{56}
Method 2: DESX

\[ E : K \times \{0,1\}^n \rightarrow \{0,1\}^n \] 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 \( 2^{64+56} = 2^{120} \)

Note: \( k_1 \oplus E(k_2, m) \) and \( E(k_2, m \oplus k_1) \) does nothing!
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 $k$

$\Rightarrow$ never implement crypto primitives yourself ...
Block ciphers

AES – Advanced encryption standard
The AES process

• 1997: DES broken by exhaustive search
• 1997: NIST publishes request for proposal
• 1998: 15 submissions
• 1999: NIST chooses 5 finalists
• 2000: NIST chooses Rijndael as AES
  (developed by Daemen and Rijmen at K.U. Leuven, Belgium)

Key sizes: 128, 192, 256 bits
Block size: 128 bits
AES core idea: Subs-Perm network

DES is based on Feistel networks

AES is based on the idea of

*substitution-permutation networks*

That is, alternating steps of substitution and permutation operations
AES: Subs-Perm network

input

$S_1$
$S_2$
$S_3$
$S_8$

subs. layer

perm. layer

$\oplus$ $k_1$

output

$S_1$
$S_2$
$S_3$
$S_8$

$\oplus$ $k_2$

$\ldots$

$S_1$
$S_2$
$S_3$
$S_8$

$\oplus$ $k_n$

inversion
AES128 schematic

Key expansion: 16 bytes → 176 bytes
The round function

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

• **ShiftRows**:

\[
\begin{array}{cccc}
  s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\
  s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\
  s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\
  s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \\
\end{array}
\]

• **MixColumns**:

\[
\begin{array}{cccc}
  s_{0,0} & s_{0,c} & s_{0,2} & s_{0,3} \\
  s_{1,0} & s_{1,c} & s_{1,2} & s_{1,3} \\
  s_{2,0} & s_{2,c} & s_{2,2} & s_{2,3} \\
  s_{3,0} & s_{3,c} & s_{3,2} & s_{3,3} \\
\end{array}
\]

\[
\begin{array}{cccc}
  s'_{0,0} & s'_{0,c} & s'_{0,2} & s'_{0,3} \\
  s'_{1,0} & s'_{1,c} & s'_{1,2} & s'_{1,3} \\
  s'_{2,0} & s'_{2,c} & s'_{2,2} & s'_{2,3} \\
  s'_{3,0} & s'_{3,c} & s'_{3,2} & s'_{3,3} \\
\end{array}
\]
## Code size/performance tradeoff

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

- Pre-compute round functions: The code size is larger (24KB or 4KB) but provides faster performance due to table lookups and xors.
- Pre-compute S-box only: The code size is smaller (256 bytes) but is slower compared to the pre-compute round functions.
- No pre-computation: The code size is smallest but the performance is slowest.
Size + performance (Javascript AES)

AES in the browser

(1) Uncompress code (one time effort)
(2) Pre-compute tables (one time effort)
(3) Perform encryption using tables

Implementation: Stanford Javascript Crypto Library
https://crypto.stanford.edu/sjcl/
Modes of operation
Electronic Code Book (ECB) Mode

Problem:

\[ m_1 = m_2 \implies c_1 = c_2 \]
What can possibly go wrong?

Plaintext

Ciphertext

Images from Wikipedia
Semantic security for ECB mode

ECB is not semantically secure for messages that contain more than one block

\[ \text{Adv}_{SS}[A,\text{ECB}] = \]

Challenger

\[ k \leftarrow K \]

Adversary A

\[ \text{c_1, c_2} \leftarrow E(k, m_b) \]

Two blocks

\[ m_0 = \text{"Hello World"} \]

\[ m_1 = \text{"Hello Hello"} \]

\[ b \in \{0, 1\} \]
Deterministic counter mode

from a PRF \( F \):
\[
E_{\text{DETCTR}}(k,m) = \oplus \begin{array}{cccc}
F(k,0) & F(k,1) & F(k,2) & \cdots & F(k,L) \\
\end{array}
\]

Stream cipher built from a PRF (e.g. AES, 3DES)
Better than ECB but only works as long as the key is only used once (one-time-key)
Semantic security under CPA

Modes that return the same ciphertext (e.g., ECB, CTR) for the same plaintext are not semantically secure under a chosen plaintext attack (CPA) (many-time-key)

$$b \in \{0, 1\}$$

Challenger

$$k \leftarrow K$$

Adversary A

$$m_0, m_1 \in M$$

$$C_0 \leftarrow E(k, m)$$

$$m_0, m_1 \in M$$

$$C_b \leftarrow E(k, m_b)$$

Encryption modes must be randomized or use a nonce (or are vulnerable to CPA)
Semantic security under CPA

Modes that return the same ciphertext (e.g., ECB, CTR) for the same plaintext are not semantically secure under a chosen plaintext attack (CPA) (many-time-key)

Two solutions:

1. Randomized encryption
   Encrypting the same msg twice gives different ciphertexts (w.h.p.)
   Ciphertext must be longer than plaintext

2. Nonce-based encryption
Nonce-based encryption

Nonce $n$: a value that changes for each msg.

$E(k,m,n) / D(k,c,n)$

$(k,n)$ pair never used more than once
Nonce-based encryption

Method 1: Nonce is a counter

Used when encryptor keeps state from msg to msg
If decryptor has same state, nonce need not be transmitted (i.e., $\text{len}(\text{PT}) = \text{len}(\text{CT})$)

Method 2: Sender chooses a random nonce

No state required but nonce has to be transmitted with CT
Cipher block chaining mode (CBC)

Let (E,D) be a PRP. \( E_{\text{CBC}}(k,m) \): chose random \( IV \in X \) and do:

\[
\begin{align*}
\text{IV} & \quad \oplus \quad m[0] & \quad \oplus \quad m[1] & \quad \oplus \quad m[2] & \quad \oplus \quad m[3] \\
E(k,\cdot) & \quad E(k,\cdot) & \quad E(k,\cdot) & \quad E(k,\cdot) \\
\text{IV} & \quad c[0] & \quad c[1] & \quad c[2] & \quad c[3]
\end{align*}
\]

Decryption:
\[
c[0] = E(k, \text{IV} \oplus m[0]) \rightarrow m[0] = \text{ciphertext}
\]
Attack on CBC with Predictable IV

Suppose given $c \leftarrow E_{CBC}(k,m)$ Adv. can predict IV for next msg.

$b \in \{0, 1\}$

Challenger

$k \leftarrow K$

Adversary A

$m_0 = IV \oplus IV_1$, $m_1 \neq m_0 \in M$

$c \leftarrow [IV, E(k,0 \oplus IV_1)]$

$c \leftarrow [IV, E(k,m_1 \oplus IV)]$

$0 \in X$

Bug in SSL/TLS 1.1: IV for record #i is last CT block of record #(i-1)
Nonce-based CBC

CBC with unique nonce: key = (k, k₁) two independent keys
unique nonce means: (key,n) pair is used for only one msg.

\[ \text{nonce} \] \quad \text{m[0]} \quad \text{m[1]} \quad \text{m[2]} \quad \text{m[3]} \]

\[ \begin{align*}
\text{IV} & \quad \oplus \quad \oplus \quad \oplus \quad \oplus \\
\text{E}(k₁, \cdot) & \quad \text{E}(k, \cdot) \quad \text{E}(k, \cdot) \quad \text{E}(k, \cdot) \quad \text{E}(k, \cdot) \\
\text{nonce} & \quad \text{c[0]} \quad \text{c[1]} \quad \text{c[2]} \quad \text{c[3]} \quad \text{ciphertext} \\
\end{align*} \]

Included only if unknown to decryptor
CBC: padding

TLS: for $n > 0$ $n$ byte pad is: $\text{n n ... n}$
If no pad needed, add a dummy block: $\text{16 16 ... 16}$

Padding oracle side channel attacks
Cipher block chaining mode (CBC)

Example applications:

1. File system encryption:
   use the same AES key to encrypt all files (e.g., loopaes)

2. IPsec:
   use the same AES key to encrypt multiple packets

Problem:

If attacker can predict IV, CBC is not CPA-secure
Summary

Block ciphers

– Map fixed length input blocks to same length output blocks
– Canonical block ciphers: 3DES, AES
– PRPs are effectively block ciphers
– PRPs can be created from arbitrary functions through Feistel networks
  • 3DES based on Feistel networks
  • AES based on substitution-permutation networks
Questions?
Linear and differential attacks \[\text{[BS’89,M’93]}\]

Given many inp/out pairs, can recover key in time less than \(2^{56}\).

**Linear cryptanalysis** (overview):

let \(c = \text{DES}(k, m)\)

Suppose for random \(k, m\):

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

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

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

Thm: given \(1/\varepsilon^2\) random \((m, c=\text{DES}(k, m))\) pairs then

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

with prob. \(\geq 97.7\%\)

\(\Rightarrow\) with \(1/\varepsilon^2\) inp/out pairs can find \(k[l_1, ..., l_u]\) in time \(\approx 1/\varepsilon^2\).
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 linearity 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

\( \text{time } O(\sqrt{|K|}) \)

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

quantum computer \( \Rightarrow \) 256-bits key ciphers
(e.g. AES-256)