Blockcipher

Viet Tung Hoang

The slides are loosely based on those of Prof. Mihir Bellare, UC San Diego.
Agenda

1. Blockcipher and Key Recovery

2. A Bird’s-Eye View of Real Blockciphers
Blockcipher

\[ E \colon \{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n \]

Key space \hspace{1cm} Domain

efficiently invertible given the key
Random key $K$ is known to both parties, but not given to adversary $A$
Key-Recovery Attack: Scenario

\[ C_1 \leftarrow E_K(M_1) \quad \ldots \quad C_q \leftarrow E_K(M_q) \]

Guess \( K \)
Modeling Key-Recovery Attack

**Game** $\text{KR}_E$

- **procedure** Initialize()
  
  $K \leftarrow \mathcal{K}$

- **procedure** Enc($M$)
  
  return $E_K(M)$

- **procedure** Finalize($K'$)
  
  return ($K' = K$)

Multiple chosen-plaintext queries

\[ \text{Adv}^\text{kr}_E(A) = \Pr[\text{KR}_E \Rightarrow 1] \]

\[ \text{Adv}^\text{kr}_E(A) \approx 0 \text{ means } A \text{ is doing poorly} \]
Practicing Key-Recovery Attack

\[ E_K(M) = M \oplus K \]

\[ E_K(M) = \pi(M \oplus K) \]

\( \pi, \pi^{-1} \) are public
Agenda

1. Blockcipher and Key Recovery

2. A Bird’s-Eye View of Real Blockciphers
DES: Parameters and History

- Designed by IBM in 1974
- Used in ATM machines
- Replaced in 2001
Design of DES: Feistel Network

(One-round) Feistel($K, \cdot$)

Inverse of Feistel

**Question:** How to invert?
Construction of DES

- Key scheduler
- $K$
- $K_1$, ..., $K_{16}$

Unkeyed processing

- $M$
- $\text{Process}(\cdot)$
- $\text{Feistel}(K_1, \cdot)$
- $\cdots$
- $\text{Feistel}(K_{16}, \cdot)$
- $\text{Process}^{-1}(\cdot)$

$C$
Exhaustive Key Search Attack

For \( K \in \mathcal{K} \) do

If \( E_K(M_i) = C_i \) for every \( i \in \{1, \ldots, d\} \) then return \( K \)

For \( E : \{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n \), if \( d > k/n \) then \( \text{Adv}^{kr}_E(A) \approx 1 \)
Exhaustive Key Search Attack on DES

<table>
<thead>
<tr>
<th>Timeline</th>
<th>Source</th>
<th>Attack time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1997</td>
<td>DESCHALL</td>
<td>96 days</td>
</tr>
<tr>
<td>1998</td>
<td>Distributed.net</td>
<td>41 days</td>
</tr>
<tr>
<td>1998</td>
<td>EFF</td>
<td>56 hours</td>
</tr>
<tr>
<td>1998</td>
<td>Distributed.net + EFF</td>
<td>22 hours</td>
</tr>
</tbody>
</table>
Incorrect Fix of DES: Double Encryption

112-bit key $\rightarrow$ prohibitive for exhaustive key search

But there’s a more clever attack!
Meet-in-the-Middle Attack: Intuition

\[ E_{K_1}(M) \]  \[ E_{K_2}^{-1}(C) \]
Meet-in-the-Middle Attack

Let $L_1, \ldots, L_N$ be all possible DES keys

\[ N = 2^{56} \]

<table>
<thead>
<tr>
<th>$L_1$</th>
<th>$E_{L_1}(M)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_i$</td>
<td>$E_{L_i}(M)$</td>
</tr>
<tr>
<td>$L_N$</td>
<td>$E_{L_N}(M)$</td>
</tr>
</tbody>
</table>

Find $L_i, L_j$ such that $E_{L_i}(M) = E_{L_j}^{-1}(C)$
How to implement in linear time?
There are on average $2^{48}$ pairs of matching keys.
How to eliminate false positives?
The 3DES Constructions

3DES2

$E_{K_1} \rightarrow E_{K_2}^{-1} \rightarrow E_{K_1}$

3DES3

$E_{K_1} \rightarrow E_{K_2}^{-1} \rightarrow E_{K_3}$
Block Size Matters, Too

\[ M \rightarrow E \rightarrow C \]

- \( k \) bits
- \( n \) bits

Distinguish outputs from random

**Birthday attack:** \( O(2^{n/2}) \) time

**Practical** for DES/2DES/3DES

\( n = 64 \)
State of the Art: AES

- NIST standard since 2001
- Best known key-recovery attack takes about $2^{126}$ time

$k \in \{128, 192, 256\}$
Security Against Key Recovery Is Not Enough

A Bad Example: Consider the following $E : \{0, 1\}^{128} \times \{0, 1\}^{256} \rightarrow \{0, 1\}^{256}$

$E_K(M_1 M_2) = AES_K(M_1) \parallel M_2$

As secure against key recovery as AES
Send half of the message in the clear!
So What Is a Good Blockcipher?

<table>
<thead>
<tr>
<th>Possible Properties</th>
<th>Necessary</th>
<th>Sufficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security against key recovery</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Hard to find $M$ given $C \leftarrow E_K(M)$</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Want:** a single “master” property that is sufficient to ensure security of common usage of blockcipher.