Block Ciphers: DES, AES

Guevara Noubir
http://www.cs.uci.edu/~noubir/Courses/CS253/2012
Reading: Chapter 3

Outline

1. Substitution-Permutation Networks
   a. Linear Cryptanalysis
   b. Differential Cryptanalysis
   c. DES
   d. AES
   e. Modes of Operation

Block Ciphers

Typical design approach:
   a. Product cipher: substitutions and permutations
      - Leading to a non-idempotent cipher
   b. Iteration:
      - Ri: number of rounds
      - Key schedule: k \rightarrow k_1, k_2, ..., k_R
      - Subkeys derived according to publicly known algorithm
      - w: state
   c. Round function
      - w^i = g(w^i-1, k_i)
      - w^0: plaintext x
      - Required property of g: ?
   d. Encryption and Decryption sequence
SPN: Substitution Permutation Networks

- SPN: special type of iterated cipher (wi/ small change)
  - Block length: $l \times m$
  - $x = x_0, x_1, \ldots, x_{90}$
  - Components:
    - Substitution cipher $s$: $(0, 1)^l \rightarrow (0, 1)^l$
    - Permutation cipher (5-box) $p$: $(1, \ldots, 15) \rightarrow (1, \ldots, 15)$
  - Outline:
    - Iterate $N$ times: $m$ substitutions; 1 permutation; $k$ sub-key;
- Definition of SPN cryptosystems:
  - $P = \gamma; C = \gamma; K \subseteq \gamma$
  - Algorithm:
    - Designed to allow decryption using the same algorithm
    - What are the parameters of the decryption algorithm?

---

SPN: Example

- $l = m = 4; N_r \equiv 4$
- Key schedule:
  - $k: (x_0, \ldots, x_{112})$
  - $K: (x_0, \ldots, x_{112})$

---

Linear Cryptanalysis

- Assumption:
  - Assume there exists a probabilistic linear relationship between a subset of plaintext bits and a subset of state bits immediately preceding the last substitution
  - Attacker has a large amount of plaintext-ciphertext encrypted using the same key
- Principle:
  - Consider a set of potential sub-keys, whenever a sub-key verifies the linear relation, increment its counter
  - The sub-key best matching probability could contain the correct values of key bits
Piling-up Lemma
Given:
- $X_1, X_2, \ldots$ independent random variables
- $\Pr[X_1 = 0] = p_1; \Pr[X_1 = 1] = 1 - p_1, 0 \leq p_1 \leq 1$
- Let $e = p_1 - 1/2$ denote the bias of the distribution

Piling-up Lemma:
- Let $e_{X_1 \oplus \ldots \oplus X_k}$ denote the bias of the random variable $X_1 \oplus X_2 \oplus \ldots \oplus X_k$. Then:
  $$e_{X_1 \oplus \ldots \oplus X_k} = 2^{-k} \sum_{i=1}^{k} e_i$$

Corollary:
- If $e_i = 0$ for one variable $\Rightarrow e_{X_1 \oplus \ldots \oplus X_k} = 0$

Linear Approximations of S-boxes

S-box:
- $I_2^2((0, 1)^n) \rightarrow (0, 1)^n$
- Input:
  - $X = (X_1, X_2, \ldots, X_n)$
  - Each coordinate defines a random variable $X_i$ with bias $0$
- Variables $X_1$ are independent
- Output:
  - $Y = (Y_1, Y_2, \ldots, Y_n)$
  - Variables $Y_1$ are not independent from each other and $X_1$
  - If $(y_1, \ldots, y_n) = I_2^2((x_1, \ldots, x_n))$ and $x_1 = x_2 = \ldots = x_n$
  - Then $Y_1 = Y_2 = \ldots = Y_n = 1$
- Therefore: one can compute the bias of: $X_1 \oplus \ldots \oplus X_n \oplus Y_1 \oplus \ldots \oplus Y_n$

Example

<table>
<thead>
<tr>
<th>$P$ Table</th>
<th>$I_2$ Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1$</td>
<td>$X_2$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- $\Pr[X_1 \oplus X_2 \oplus Y_1 = 0] = ?$
- $\Pr[X_1 \oplus X_2 \oplus Y_2 = 0] = ?$
- In general one computes the biases for all possible subsets:
  $$\left( \bigoplus_{i=1}^{m} X_i \right) \oplus \left( \bigoplus_{i=1}^{m} Y_i \right)$$
- Derive a table $(a, b) \Rightarrow$ Number of times we get a 0; $N_{(a, b)}$
  | Entry for $a=3$, $b=9$ | Bias $c(a, b) = (N_{(a, b)} - b)/16$ |
|----------------|----------------|----------------|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |

- $N_{3, 9} = 2$
- $N_{2, 4} = 2$
Linear Attack on SPN

Approach:
- Find a set of linear approximations of S-boxes that can be used to derive a linear approximation of the SPN (excluding the last round)
- Derive the bias value using piling-up lemma
- This linear approximation depends on some subset of the key bits
- For each possible pair (plaintext, ciphertext), and each key increment the counter of the subkey if the approximation equation gives a 0
- Hopefully the correct value for sub-keys will have the expected bias

Example

Approximation equations:
- $T_1 = U^1 \oplus U^1 \oplus U^2 \oplus U^2$
- $T_2$ has bias -1/4
- $T_3$ has bias -1/4
- $T_4$ has bias -1/4
- $T_5 = U^3 \oplus V^3 \oplus V^4$

If $T_0$, $T_5$, $T_6$, $T_7$ were independent then the bias of $T_1 \oplus T_2 \oplus T_3 \oplus T_4$ would be -1/32

We will make this non-rigorous approximations, because it seems to work in practice

$T_1 \oplus T_2 \oplus T_3 \oplus T_4 = X_0 \oplus X_0 \oplus X_0 \oplus V^1 \oplus V^1 \oplus V^1 \oplus V^1 \oplus K^1 \oplus K^1 \oplus K^1 \oplus K^1 \oplus K^2 \oplus K^2 \oplus K^2 \oplus K^2 \oplus K^3 \oplus K^3 \oplus K^3 \oplus K^3 \oplus K^4 \oplus K^4 \oplus K^4 \oplus K^4$
Example (Cont.)

- If the key bits in:
  - $K_0 @ K_1 @ K_0 @ K_1 @ K_1 @ K_1 @ K_1 @ K_1$ are fixed
- Then the random variable:
  - $X_1 @ X_0 @ U^n @ U^n @ U^n @ U^n$ has bias $\pm 1/32$
- This allows us to derive 8 bits for last subkey:
  - $K_{10}$ and $K_{11}$
- Outline of $\text{alg}$:
  - Build a table for all possible 256 values of $K_0$ and $K_1$
  - For each $(x, y)$ pair of plaintext and ciphertext, for each candidate subkey:
    - Obtain the values of $e_i$ and $e_j$ by decrypting $y$
    - Compute $x_i @ y_i @ x_i @ y_i @ x_i @ y_i @ x_i @ y_i$
    - If equal 0, increment counter of corresponding $K_{10}$ and $K_{11}$
  - The table entry that has bias close to 1/32 is the right value for $K_{10}$ and $K_{11}$

Requirements for Linear Cryptanalysis

- A bias of $\epsilon$ requires:
  - $T = c \cdot \epsilon^2$ pairs of plaintext-ciphertext

- For the previous example $T = 8000$ was usually successful $\Rightarrow c = 8$

Differential Cryptanalysis

- Similar to Linear Cryptanalysis:
  - Compares the XOR of two inputs to the XOR of the corresponding two outputs: $x @ y$
  - Adversary has a large number of chosen plaintext tuples $(x, x', y, y')$ s.t. $x'$ is fixed
  - Approach:
    - For each candidate key, decrypt $x$ and $y$
    - For each candidate key, compute the values of certain state bits
    - If the state bits match the most likely value for the input $x$ or then increment the candidate key counter

- Proposed Encryption Standard (PES) which is the original proposal for the International Data Encryption Standard (IDEA used in PGP) was modified to resist to this kind of attacks

- GSM A3 algorithm is sensitive to this kind of attacks
  - SIM card secret key can be recovered $\Rightarrow$ GSM cloning
Differential Cryptanalysis

- \( \Delta(x) = ((x, x' x'): x \in \{0, 1\}^n \)
- \( N_{\delta}(x', y') = |\{ (x, x'): \Delta(x) \land \Pi_{\delta}(x') = y' \}| \)
- Similarly to Linear cryptanalysis build the table:
- \( N_{\delta}(a, b) \)
- Define propagation ratio: \( R_{\delta}(a', b') = \frac{N_{\delta}(a', b')}{2^n} \)
- Differential Trail:
  - Is multi-round combination
  - Permutations and key XORs do not impact the differential
  - Sub-key recovery and filtering
- Example:
  - \( R_1(1011, 0010) = 9; R_1(0100, 0110) = 3/8; R_2(0010, 0101) = 3/8; \)
  - Propagation ratio of differential trail: 27/1024
- Complete attack requires between 50 and 100 tuples to find sub-key

Today's Block Encryption Algorithms

- Key size:
  - Two short => easy to guess
- Block size:
  - Two short easy to build a table by the attacker: (plaintext, ciphertext)
  - Reasonable size: 64 bits
- Properties:
  - One-to-one mapping
  - Mapping should look random to someone who doesn't have the key
  - Efficient to compute/reverse
- How?
  - Substitution (small chunks) & permutation (long chunks)
  - Multiple rounds
**Data Encryption Standard (DES)**

- Developed by IBM for the US government
- Based on Lucifer (64-bits, 128-bits key in 1971)
- To respond to the National Bureau of Standards CFP (now called NIST)
  - Modified characteristics (with help from NSA):
    - 64-bits block size, 56 bits key length
  - Concerns about trapdoors, key size, s-box structure
- Adopted in 1977 as the DES (FIPS PUB 46, ANSI X3.92) and reaffirmed in 1994 for 5 more years
- Today replaced by AES

```
# Feistel Cipher

**Function f** does not have to be injective!
- \( L_1 = R_0 \)
- \( R_j = L_{j-1} \oplus f(R_{j-1}, K_j) \)

How can we invert one round?
```
One DES Round

- $L_{i+1} = R_i$
- $R_{i+1} = L_i \oplus f(R_i, K_i)$

Key (56 bits)

Expansion Permutation

S-Box Substitution

P-Box Permutation

Key (56 bits)

S-Box Substitution

- 48-bit input
- S-Box heart of DES security
- S-Box: 4x6 entry table
- Input 6 bits: $A_i,A_{i+1},A_{i+2},A_{i+3}$
- 2 bits ($A_i$): determine the table row (1-4)
- 4 bits ($A_{i+1}A_{i+2}$): determine the table entry
- Output: 4 bits
- Example: $S_1(10100) = 1101$
- S- Boxes are optimized against Differential cryptanalysis

Double/Triple DES

- Double DES
  - Vulnerable to Meet-in-the-Middle Attack (DH77)

- Triple DES
  - Used two keys $K_1$ and $K_2$
  - Compatible with simple DES ($K_1=K_2$)
  - Used in ISO 8732, PEM
DES Linear/Differential Cryptanalysis

Differential cryptanalysis
- "Rediscovered" by E. Biham & A. Shamir in 1990
- Based on a chosen-plaintext attack:
  - Analyze the difference between the ciphertexts of two plaintexts which have a known fixed difference
  - The analysis provides information on the key
- 8-round DES broken with 2^{17} chosen plaintext and complexity 2^{47}
- 16-round DES requires 2^{47} chosen plaintext and complexity 2^{77}
- DES design took into account this kind of attacks
- Linear cryptanalysis
  - Uses linear approximations of the DES cipher (M. Matsui 1993)
  - Applied to DES:
    - Requires 2^n known plaintext encrypted with the same key
    - Time: 40 days to generate the pairs, 10 days to find the key

Breaking DES

Electronic Frontier Foundation built a "DES Cracking Machine" [1998]
- Attack: brute force
- Inputs: two ciphertext
- Architecture:
  - PC
  - Array of custom chips that can compute DES
  - 248 units/chip x 64kops/board x 27 boards
- Power:
  - Searches 92 billion keys per second
  - Takes 4.5 days for half the key space
  - Successfully broke "DES Challenge 0-2" in 56 hours
- Cost:
  - $130'000 (all the material, chips, boards, cooling, PC etc.)
  - $80'000 (development from scratch)

The Advanced Encryption Standard (AES)

AES = Rijndael
- Designed by Rijmen-Daemen (Belgium)
- Key size: 128/192/256 bit
- Block size: 128 bits of data
- Properties: iterative rather than Feistel cipher
  - Treats data in 4 groups of 4 bytes
  - Operates on an entire block in every round
- Designed to be:
  - Resistant against known attacks
  - Speed and code compactness on many CPUs
  - Design simplicity
AES Outline

Algorithm:
1. Initialize State ← x @ RoundKey;
2. For each of the N-1 rounds:
   1. SubBytes(State);
   2. ShiftRows(State);
   3. MixColumns(State);
   4. AddRoundKey(State);
3. Last round:
   1. SubBytes(State);
   2. ShiftRows(State);
   3. AddRoundKey(State);
4. Output y ← State

Finite Fields - Break

- $GF(2) = Z_2$
- $GF(2^n) = \{ (a_0, a_1, ..., a_{n-1}) | a_i \in GF(2) \}$
  - Addition can be carried out bit by bit
  - Multiplication mod a primitive polynomial
  - Generation of $GF(2^n)$: done by polynomial modulo a primitive polynomial of degree $n$, $m(x)$
  - Elements of $GF(2)$ can be represented as a polynomial $(a_0, a_1, ..., a_{n-1}) = a(x) = a_0 + a_1x + ... + a_{n-1}x^{n-1}$
  - Textbook Notation: $F_{2^n} = Z_2[x]/m(x)$
  - AES uses $n=8$, and $m(x) = x^8 + x^4 + x^3 + x + 1$

Primitive Polynomial

- A polynomial $f(x)$ over a field $Q$ is said to be irreducible if $f(x)$ cannot be factored over $Q$
- A polynomial $f(x)$ over a field $Q$ is said to be primitive if every root of $f(x)$ generates the extension field
  - Over $GF(2)$ a polynomial is primitive if the smallest $k$ for which $f(x)$ divides $x^k - 1$ is $k = 2^n - 1$
- Example,
  - $f(x) = x^4 + x^3 + x + 1$ over $GF(2)$
  - $f(x)$ is irreducible but not primitive
  - $g(x) = x^2 + x + 1$
  - $g(x)$ is primitive
Test Irreducibility

To show $x^8 + x^4 + x^3 + x + 1$ is irreducible:
- If the number of terms is odd over $GF(2)$, then it cannot be divisible by $x+1$
- Try dividing by polynomials of degree 2, $x^2 + x + 1$
- Try polynomials of degree 3, $x^3 + x + 1$ and $x^3 + x^2 + 1$
- Try polynomials of degree 4, $x^4 + x^3 + x^2 + x + 1$, $x^4 + x^3 + 1$,
  $x^4 + x^3 + 1$, $x^4 + x + 1$
- Do not require any more testing beyond degree 4

Test Primitivity

To show $x^8 + x^4 + x^3 + x + 1$ is not primitive:
- Take $\alpha$ to be a root of the polynomial, that is,
  $\alpha^4 = \alpha^3 + \alpha^2 + \alpha + 1$
- $\alpha^5 = \alpha^4 + \alpha^3 + \alpha = \alpha^4 + \alpha^2 + 1 + \alpha^2 + \alpha$
  $\alpha = 1$
- To show $x^8 + x^4 + x^3 + x + 1$ is primitive
  - Take $\alpha$ to be a root of the polynomial, that is,
    $\alpha^8 = \alpha^6 + \alpha^3 + \alpha + 1$ ($00110110$) = 1b
    $\alpha^9 = \alpha^8 + \alpha^4 + \alpha$ ($00110110$) = 36
    $\alpha^{12} = \alpha^9 + \alpha^5 + \alpha + 1$ ($10101011$) = ab

Multiplication in $GF(2^8)$

$\alpha^9 \cdot \alpha^{12} = \alpha^{9+12} = \alpha^{21} \mod 127$

$36 \cdot (ab) = (00110110) \cdot (10101011) = 11110010$
AES (Continued)

- State: 16 bytes structured in an array

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

- Each byte is seen as an element of \( \mathbb{F}_2 = \text{GF}(2^8) \)
- \( \mathbb{F}_2 \): finite field of 256 elements
  - Operations:
    - Elements of \( \mathbb{F}_2 \) are viewed as polynomials of degree 7 with coefficients \((0, 1)\)
    - Addition: polynomials addition \( \oplus \)
    - Multiplication: polynomials multiplication modulo \( x^8 + x^4 + x^3 + 1 \)

---

SubBytes

- **Input:** \( a = (b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0) \)
- **Output:** \( b = (b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0) \)

If \( a = 0 \) then \( b = a^2 \):

\[
\begin{array}{cccccccc}
b_0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
b_1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
b_2 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\
b_3 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
b_4 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
b_5 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\
b_6 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\
b_7 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
\end{array}
\]

SubBytes is non-linear
Example: \( \text{SubBytes}(0x53) = 0x7c \); \( 0x53^2 = 0xca \)
SubBytes can also be defined as a 16x16 table

---

ShiftRows

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

---
MixColumn

- Input:
  - c: column index
  - s: state
- Output:
  - s: new state
- For i = 0 to 3
  1. \( s_i = s_{i-1} \oplus s_i \oplus s_{i+1} \oplus s_{i+2} \)
- Note:
  - Multiplications are in \( \mathbb{F}_2 \)

Key Expansion

- 10-round case (128 bits key)
  - \( RCon[1-10] \leftarrow \text{CONSTANTS} \)
  - For i = 0 to 3
    - do \( w[i] \leftarrow \text{Rev}(w[i]), \text{key}(4i+1), \text{key}(4i+2), \text{key}(4i+3) \)
      // \( w \) has 4 blocks of 32 bits
  - For i = 4 to 43
    - do temp \( \leftarrow w[i] \)
      if \( i = 0 \pmod{4} \) then temp \( \leftarrow \text{SubWord}(\text{RevWord}(\text{temp})) \odot RCon[i/4] \)
      \( w[i] \leftarrow w[i-1] \oplus \text{temp} \)
    - Return \( (w[0], ..., w[43]) \)

Implementation Aspects

- Can be efficiently implemented on a 8-bit CPU
  - Byte substitution works on bytes using a table of 256 entries
  - Shift rows is simple byte shifting
  - Add round key works on byte XORs
  - Mix columns requires matrix multiply in \( \mathbb{F}_2 \), which works on byte values, can be simplified to use a table lookup
Implementation Aspects

- Can be efficiently implement on 32-bit CPU
  - Redefine steps to use 32-bit words
  - Can pre-compute 4 tables of 256-words
  - Then each column in each round can be computed using 4 table lookups + 4 XORs
  - At a cost of 16kB to store tables
  - See Problem Set 4.
- Designers believe this very efficient implementation was a key factor in its selection as the AES cipher

Modes of Operation DES/AES: Electronic Codebook (ECB)

Cipher Block Chaining (CBC)
**Cipher Feedback (CFB)**

**DES Modes:**

**Output Feedback (OFB)**

**Counter (CTR)**

- A "new" mode, though proposed early on
- Similar to OFB but encrypts counter value rather than any feedback value
- Must have a different key & counter value for every plaintext block (never reused)
  
  \[ C_i = P_i \oplus O_i \]
  
  \[ O_i = \text{DES}_{K_1}(i) \]

- Uses:
  - High-speed network encryptions