Block Ciphers

CS 6750     Lecture 3

September 24, 2009

Riccardo Pucella
Product Cryptosystems

A way to combine cryptosystems
For simplicity, assume endomorphic cryptosystems
I.e., where $C=P$

$S_1 = (P, P, K_1, E_1, D_1)$
$S_2 = (P, P, K_2, E_2, D_2)$

Product cryptosystem $S_1 \times S_2$ is defined to be
$(P, P, K_1 \times K_2, E, D)$

where

$e_{(k_1,k_2)}(x) = e_{k_2}(e_{k_1}(x))$
$d_{(k_1,k_2)}(y) = d_{k_1}(d_{k_2}(y))$
Product Cryptosystems

If $\Pr_1$ and $\Pr_2$ are probability distributions over the keys of $S_1$ and $S_2$ (resp.)

Take $\Pr$ on $S_1 \times S_2$ to be $\Pr(<k_1,k_2>) = \Pr_1(k_1)\Pr_2(k_2)$

That is, keys are chosen independently

Some cryptosystems commute, $S_1 \times S_2 = S_2 \times S_1$

Some cryptosystems can be decomposed into $S_1 \times S_2$

Affine cipher can be decomposed into $S \times M = M \times S$

(Some subtleties about key probabilities matching)
Idempotence

A cryptosystem is **idempotent** if $S \times S = S$

E.g. shift cipher, substitution cipher, Vigenère cipher...

(Again, subtleties about key probabilities matching)

An idempotent cryptosystem does not gain additional security by iterating it

But iterating a nonidempotent cryptosystem does!
A Nonidempotent Cryptosystem

1. Fix $m > 1$

2. Let $S_{\text{sub}}$ a substitution cipher over $(\mathbb{Z}_{26})^m$

3. Let $S_{\text{prm}}$ be the permutation cipher:
   - $C = P = (\mathbb{Z}_{26})^m$
   - $K = \{ \pi : \pi \text{ a permutation } \{1,\ldots,m\} \to \{1,\ldots,m\} \}$
   - $e_{\pi}(<x_1, \ldots, x_m>) = <x_{\pi(1)}, \ldots, x_{\pi(m)}>$
   - $d_{\pi}(<y_1, \ldots, y_m>) = <y_{\eta(1)}, \ldots, y_{\eta(m)}>$, where $\eta = \pi^{-1}$

4. Theorem: $S_{\text{sub}} \times S_{\text{prm}}$ is not idempotent
Iterated Cryptosystems

- A kind of product cryptosystem

- Idea: given S a cryptosystem, an iterated cryptosystem is $S \times S \times \ldots \times S = S^N$
  - $N = \text{number of iterations} = \text{rounds}$
  - A key is a tuple $<k_1, \ldots, k_N>$
    - $k_i = \text{key for round } i = \text{round key}$
  - Only useful if S is not idempotent

- Generally, the key is derived from an initial key k
  - k is used to derive $<k_1, \ldots, k_N>$ (= key schedule)
  - Derivation is via a fixed and known algorithm
Iterated Cryptosystems

Iterated cryptosystems are often described using a round function \( g : P \times K \rightarrow C \)

\( g (w, k) \) gives the encryption of \( w \) using round key \( k \)

To encrypt \( x \) using key schedule \( <k_1, \ldots, k_N> \):

\[
\begin{align*}
w_0 & \leftarrow x \\
w_1 & \leftarrow g (w_0, k_1) \\
w_2 & \leftarrow g (w_1, k_2) \\
& \ldots \\
w_N & \leftarrow g (w_{N-1}, k_N) \\
y & \leftarrow w_N
\end{align*}
\]
Iterated Ciphers

- To decrypt, require $g$ to be **invertible** when round key is fixed
  - I.e., there exists $g^{-1}$ such that $g^{-1}(g(w, k), k) = w$
  - Requires $g$ to be injective in its first argument

- To decrypt ciphertext $y$ using key schedule $<k_1, ..., k_N>$
  
  $w_N \leftarrow y$
  $w_{N-1} \leftarrow g^{-1}(w_N, k_N)$
  $w_{N-2} \leftarrow g^{-1}(w_{N-1}, k_{N-1})$
  ...
  $w_0 \leftarrow g^{-1}(w_1, k_1)$
  $x \leftarrow w_0$
Substitution-Permutation Networks

- A special case of iterated cryptosystem
  - Foundation for DES and AES

- Plaintext/ciphertext: binary vectors of length $l \times m$
  - $(\mathbb{Z}_2)^{lm}$

- Substitution $\pi_S : (\mathbb{Z}_2)^l \rightarrow (\mathbb{Z}_2)^l$
  - Replace $l$ bits by new $l$ bits
  - Often called an S-box
  - Creates confusion

- Permutation $\pi_P : (\mathbb{Z}_2)^{lm} \rightarrow (\mathbb{Z}_2)^{lm}$
  - Reorder $lm$ bits
  - Creates diffusion
Substitution-Permutation Networks

- **N rounds**
- Assume a key schedule for key $k = <k_1, ..., k_{N+1}>$
  - Don’t care how it is produced
  - Round keys have length $l \times m$

Write string $x$ of length $l \times m$ as $x_{<1>} || ... || x_{<m>}$
- Where $x_{<i>} = <x_{(i-1)l+1}, ..., x_{il}>$ of length $l$

At each round but the last:
1. Add round key bits to $x$
2. Perform $\pi_S$ substitution to each $x_{<i>}$
3. Apply permutation $\pi_P$ to result

Permutation not applied on the last round
- Allows the “same” algorithm to be used for decryption
Substitution-Permutation Networks

Algorithmically (with key schedule \(<k_1, \ldots, k_{N+1}>\):

\[
\begin{align*}
 w_0 & \leftarrow x \\
 \text{for } r & \leftarrow 1 \text{ to } N-1 \\
 u^r & \leftarrow w_{r-1} \oplus k_r \\
 \text{for } i & \leftarrow 1 \text{ to } m \\
 v^r_{<i>} & \leftarrow \pi_S (u^r_{<i>}) \\
 w_r & \leftarrow <v^r_{\pi_P(1)}, \ldots, v^r_{\pi_P(l \times m)}> \\
 u^N & \leftarrow w_{N-1} \oplus k_N \\
 \text{for } i & \leftarrow 1 \text{ to } m \\
 v^N_{<i>} & \leftarrow \pi_S (u^N_{<i>}) \\
 y & \leftarrow v^N \oplus k_{N+1}
\end{align*}
\]
Example

Stinson, Example 3.1

\( l = m = N = 4 \)

So plaintexts are 16 bits strings

Fixed \( \pi_S \) that substitutes four bits into four bits

Table: E,4,D,1,2,F,B,8,3,A,6,C,5,9,0,7 (in hexadecimal!)

Fixed \( \pi_P \) that permutes 16 bits

Perm: 1,5,9,13,2,6,10,14,3,7,11,15,4,8,12,16

Key schedule:

Initial key: 32 bits key \( K \)

Round \( r \) key: 16 bits of \( K \) from positions 1, 5, 9, 13
Comments

- We could use different S-boxes at each round
- Example not very secure
  - Key space too small: $2^{32}$
- Could improve:
  - Larger key size
  - Larger block length
  - More rounds
  - Larger S-boxes
break
Feistel Cryptosystems

A special case of iterated cryptosystems

At each round, string is divided equally into L and R

Round function $g$ takes $L_{i-1}R_{i-1}$ and $K_i$, and returns a new string $L_iR_i$ given by:

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

To decrypt, use inverse of $g$:

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

**OBSERVATION**: $f$ need not be invertible!
DES

“Data Encryption Standard”

Developed by IBM, from an earlier cryptosystem Lucifer

 Adopted as a standard for “unclassified” data: 1977

16 round Feistel cryptosystem:
- encrypts 64 bits vectors
DES Key Schedule

- Initial key: 64 bits
  - Only 56 bits of the key are used
  - every 8th bit is a parity bit to ensure no error in transmission
  - the 8th bit is set to 0 or 1 to make the number of 1's in the full 8 bits odd.

- Key schedule:
  - 56 bits key \( k \) produces \(<k_1, ..., k_{16}>\), 48 bits each
  - Round keys obtained by permutation of selection of bits from key \( k \)
  - (Details in the handout)
DES Encryption/Decryption

To encrypt plaintext $x$:
1. Apply fixed permutation $IP$ to $x$ to get $L_0R_0$
2. Do 16 rounds of DES
3. Apply fixed permutation $IP^{-1}$ to get ciphertext

(Permutation IP motivated by hardware considerations)

To decrypt ciphertext $y$:
1. Apply fixed permutation $IP$ to $y$ to get $L_{16}R_{16}$
2. Do 16 “inverse” rounds of DES
3. Apply fixed permutation $IP^{-1}$ to get plaintext
DES Round

To describe a round of DES, need to give function $f$

Takes string $A$ of 32 bits and a round key $J$ of 48 bits

Computing $f(A, J)$:

1. Expand $A$ to 48 bits via fixed expansion $E(A)$
2. Compute $E(A) \oplus J = B_0 B_1 ... B_8$ (each $B_i$ is 6 bits)
3. Use 8 fixed S-boxes $S_1, ..., S_8$, each $\{0,1\}^6 \rightarrow \{0,1\}^4$
   Get $C_i = S_i(B_i)$
4. Set $C = C_1 C_2 ... C_8$ of length 32 bits
5. Apply fixed permutation $P$ to $C$
Linear Cryptanalysis

- Known-plaintext attack
  - Aim: find some bits of the key

- Basic idea: Try to find a linear approximation to the action of a cipher

- Can you find a (probabilistic) linear relationship between some plaintext bits and some bits of the string produced in the last round (before the last substitution)?
- If yes, then some bits occur with nonuniform probability
- By looking at a large enough number of plaintexts, can determine the most likely key for the last round
Differential Cryptanalysis

- Usually a chosen-plaintext attack
  - Aim: find some bits of the key

- **Basic idea**: try to find out how differences in the inputs affect differences in the output
  - Many variations; usually, difference $= \oplus$

- For a chosen specific difference in the inputs, can you find an expected difference for some bits in the string produced before the last substitution is applied?
  - If yes, then some bits occur with nonuniform probability
  - By looking at a large enough number of pairs of plaintexts $(x_1, x_2)$ with $x_1 \oplus x_2 = \text{chosen difference}$, can determine most likely key for last round
Comments on DES

- Key space is too small
  - Can build specialized hardware to do automatic search
  - This is a known-plaintext attack

- Differential and linear cryptanalysis are difficult
  - Need $2^{43}$ plaintexts for linear cryptanalysis
  - S-boxes resilient to differential cryptanalysis

- Number of rounds is important
  - 8 rounds DES is easy to break
AES

“Advanced Encryption Standard”
- Developed in Belgium (as Rijndael)
- Adopted in 2001 as a new US standard

Iterated cryptosystem
- Block length: 128 bits
- 3 possible key lengths, with varying number of rounds
  - 128 bits (N=10)
  - 192 bits (N=12)
  - 256 bits (N=14)
High-Level View of AES

To encrypt plaintext $x$ with key schedule $<k_0, \ldots, k_N>$:

1. Initialize STATE to $x$ and add ($\oplus$) round key $k_0$

2. For first $N-1$ rounds:
   a. Substitute using S-box
   b. Permutation SHIFT-ROWS
   c. Substitution MIX-COLUMNS
   d. Add ($\oplus$) round key $k_i$

3. Substitute using S-Box, SHIFT-ROWS, add $k_N$

4. Ciphertext is resulting STATE

(Next slide describes the terms)
AES Operations

- STATE is a 4x4 array of bytes (= 8 bits)
  - Split 128 bits into 16 bytes
  - Arrange first 4 bytes into first column, then second, then third, then fourth

- S-box: apply fixed substitution \( \{0,1\}^8 \rightarrow \{0,1\}^8 \) to each cell

- SHIFT-ROWS: shift second row of STATE one cell to the left, third row of STATE two cells to the left, and fourth row of STATE three cells to the left

- MIX-COLUMNS: multiply fixed matrix with each column
AES Key Schedule

For N=10, 128 bits key
- 16 bytes: k[0], ..., k[15]
Algorithm is word-oriented (word = 4 bytes = 32 bits)
A round key is 128 bits ( = 4 words)
Key schedule produces 44 words ( = 11 round keys)
- w[0], w[1], ..., w[43]

- w[0] = <k[0], ..., k[3]>
- w[1] = <k[4], ..., k[7]>
- w[3] = <k[12], ..., k[15]>
- w[i] = w[i-4] ⊕ w[i-1]

Except at i multiples of 4 (more complex; see book)
Modes of Operation

How to use block ciphers when plaintext is more than block length

Simplest: ECB (Electronic Codebook Mode):

\[ \begin{align*}
    &X_1 \\
    &e_k \\
    &Y_1 \\
\end{align*} \quad \begin{align*}
    &X_2 \\
    &e_k \\
    &Y_2 \\
\end{align*} \quad \ldots
Modes of Operation

CFB (Cipher Feedback Mode):

\[ y_0 = IV \]

\[ e_k \xrightarrow{+} y_1 \xrightarrow{+} y_2 \xrightarrow{+} \ldots \]
Modes of Operation

CBC (Cipher Block Chaining):

\[ y_0 = IV \]
\[ y_1 = e_k \]
\[ y_2 = e_k \]
\[ y_i = X_i \]

...
Modes of Operation

OFB (Output Feedback Mode)

\[ z_0 = \text{IV} \rightarrow e_k \rightarrow z_1 \rightarrow x_1 + \rightarrow y_1 \]

\[ e_k \rightarrow z_2 \rightarrow x_2 + \rightarrow y_2 \]

...