Block Ciphers

CSG 252 Fall 2006

Riccardo Pucella

Product Ciphers

- A way to combine cryptosystems
- For simplicity, assume endomorphic cryptosystems
  - 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

$$e_{(k1,k2)}(x) = e_{k2}(e_{k1}(x))$$

$$d_{(k1,k2)}(y) = d_{k1}(d_{k2}(y))$$

Product Ciphers

- 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 $Pr(\langle k_1,k_2 \rangle) = Pr_1(k_1)Pr_2(k_2)$
  - That is, keys are chosen independently
- Some cryptosystems commute, $S_1 \times S_2 = S_2 \times S_1$
  - Not all cryptosystems commute, but some do
- Some cryptosystems can be decomposed into $S_1 \times S_2$
  - Need key probabilities to match too
  - Affine cipher can be decomposed into $S \times M = M \times S$

Product Ciphers

- A cryptosystem is idempotent if $S \times S = S$
  - Again, key probabilities must agree
  - E.g. shift cipher, substitution cipher, Vigenère cipher...
- An idempotent cryptosystem does not gain additional security by iterating it
- But iterating a nonidempotent cryptosystem does!

A Nonidempotent Cryptosystem

- Let S_{sub} the substitution cipher
- Let S_{perm} be the permutation cipher:
  - Fix $m > 1$
  - $C = P = (Z_{26})^m$
  - $K = \{ \pi \mid \pi \text{ a permutation } \{1,\ldots,m\} \rightarrow \{1,\ldots,m\} \}$
  - $e_\pi (\langle x_1, \ldots, x_m \rangle) = \langle x_{\pi(1)}, \ldots, x_{\pi(m)} \rangle$
  - $d_\pi (\langle y_1, \ldots, y_m \rangle) = \langle y_\eta(1), \ldots, y_\eta(m) \rangle$, where $\eta = \pi^{-1}$
- Theorem: $S_{sub} \times S_{perm}$ is not idempotent

Iterated Ciphers

- A form of product ciphers
- Idea: given S a cryptosystem, an iterated cipher is S×S×…×S
  - N = number of iterations (= rounds)
  - A key is of the form $\langle k_1, \ldots, k_N \rangle$
  - 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 Ciphers

- Iterated ciphers are often described using a function $g : P \times K \to C$
  - g is the round function
  - $g(w, k)$ gives the encryption of w using key k
- To encrypt x using key schedule $\langle k_1, \ldots, k_N \rangle$:

$$w_0 \leftarrow x$$
$$w_1 \leftarrow g(w_0, k_1)$$
$$w_2 \leftarrow g(w_1, k_2)$$
$$\ldots$$
$$w_N \leftarrow g(w_{N-1}, k_N)$$
$$y \leftarrow w_N$$

Iterated Ciphers

- To decrypt, require g to be invertible when key argument is fixed
  - There exists g^{-1} such that $g^{-1}(g(w, k), k) = w$
  - g injective in its first argument

- To decrypt cipher y using key schedule $\langle k_1, \ldots, k_N \rangle$

$$w_N \leftarrow y$$
$$w_{N-1} \leftarrow g^{-1}(w_N, k_N)$$
$$w_{N-2} \leftarrow g^{-1}(w_{N-1}, k_{N-1})$$
$$\ldots$$
$$w_0 \leftarrow g^{-1}(w_1, k_1)$$
$$x \leftarrow w_0$$

Substitution-Permutation Networks

- A form of iterated cipher
  - Foundation for DES and AES
- Plaintext/ciphertext: binary vectors of length l×m
  - $(Z_2)^{l \times m}$
- Substitution $\pi_S : (Z_2)^l \rightarrow (Z_2)^l$
  - Replace l bits by new l bits
  - Often called an S-box
  - Creates confusion
- Permutation $\pi_P : (Z_2)^{lm} \rightarrow (Z_2)^{lm}$
  - Reorder lm bits
  - Creates diffusion

Substitution-Permutation Networks

- N rounds
- Assume a key schedule for key $k = \langle k_1, \ldots, k_{N+1} \rangle$
  - Don't care how it is produced
  - Round keys of length $l \times m$

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

- At each round but the last:
  1. Add round key bits to x
  2. Perform π_S substitution to each $x_{\langle i \rangle}$
  3. Apply permutation π_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 $\langle k_1, \ldots, k_{N+1}\rangle$):

$$
\begin{aligned}
& w_0 \leftarrow x \\
& \text{for } r \leftarrow 1 \text{ to } N-1 \\
& \quad u^r \leftarrow w_{r-1} \oplus k_r \\
& \quad \text{for } i \leftarrow 1 \text{ to } m \\
& \quad\quad v^r_{<i>} \leftarrow \pi_S\ (u^r_{<i>}) \\
& \quad w_r \leftarrow \langle v^r_{\pi P(1)}, \ldots, v^r_{\pi P(l \times m)} \rangle \\
& u^N \leftarrow w_{N-1} \oplus k_N \\
& \text{for } i \leftarrow 1 \text{ to } m \\
& \quad v^N_{<i>} \leftarrow \pi_S\ (u^N_{<i>}) \\
& y \leftarrow v^N \oplus k_{N+1}
\end{aligned}
$$

Example

- Stinson, Example 3.1

- l = m = N = 4
  - So plaintexts are 16 bits strings

- Fixed π_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 π_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 key (round r): 16 bits of K from pos 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

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 $x_1 \oplus x_2$ = chosen difference, can determine most likely key for last round

10 minutes break

DES

- "Data Encryption Standard"
  - Developed by IBM, from Lucifer
  - Adopted as a standard for "unclassified" data: 1977

- Form of iterated cipher called a Feistel cipher
- At each round, string to be encrypted 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: $L_i = R_{i-1}$

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

- Note that f need not be invertible!
  - To decrypt: $R_{i-1} = L_i$

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

DES

- DES is a 16 round Feistel cipher
- Block length: 64 bits
- Key length: 56 bits

- 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

- Initial and final permutations do not affect security

- 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

DES Round

- To describe a round of DES, need to give function f
  - Takes string A of 32-bit 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 $E(A) \oplus J = B_0B_1...B_8$ (each B_i 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 $C = C_1C_2...C_8$ of length 48 bits
5. Apply fixed permutation P to C

Comments on DES

- Key space is too small
  - Can build specialized hardware to do automatic search
  - Known-plaintext attack
- Differential and linear cryptanalysis are difficult
  - Need 2^{43} plaintexts for linear cryptanalysis
  - S-boxes resilient to differential cryptanalysis

AES

- "Advanced Encryption Standard"
  - Developed in Belgium
  - Adopted in 2001 as a new American standard

- Iterated cipher
- Block length: 128 bits
- 3 allowed 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, ..., 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 four cells to the left
- MIX-COLUMNS: multiply fixed matrix with each column

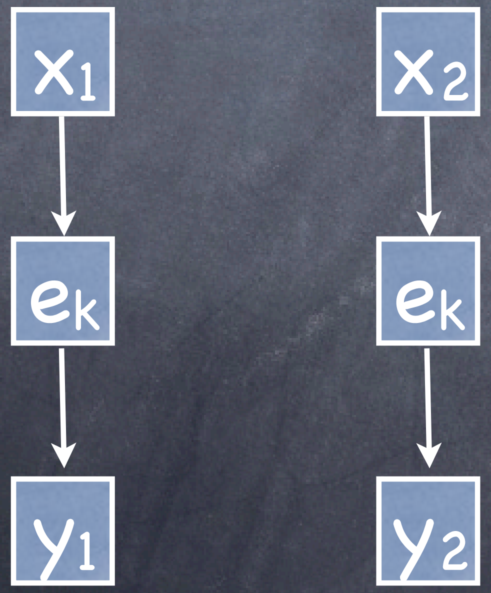
AES Key Schedule

- 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[2] = <k[8], ..., k[11]>
- w[3] = <k[12], ..., k[15]>
- w[i] = w[i-4] $\oplus$ w[i-1]
  - Except at i multiples of 4 (more complex; see book)

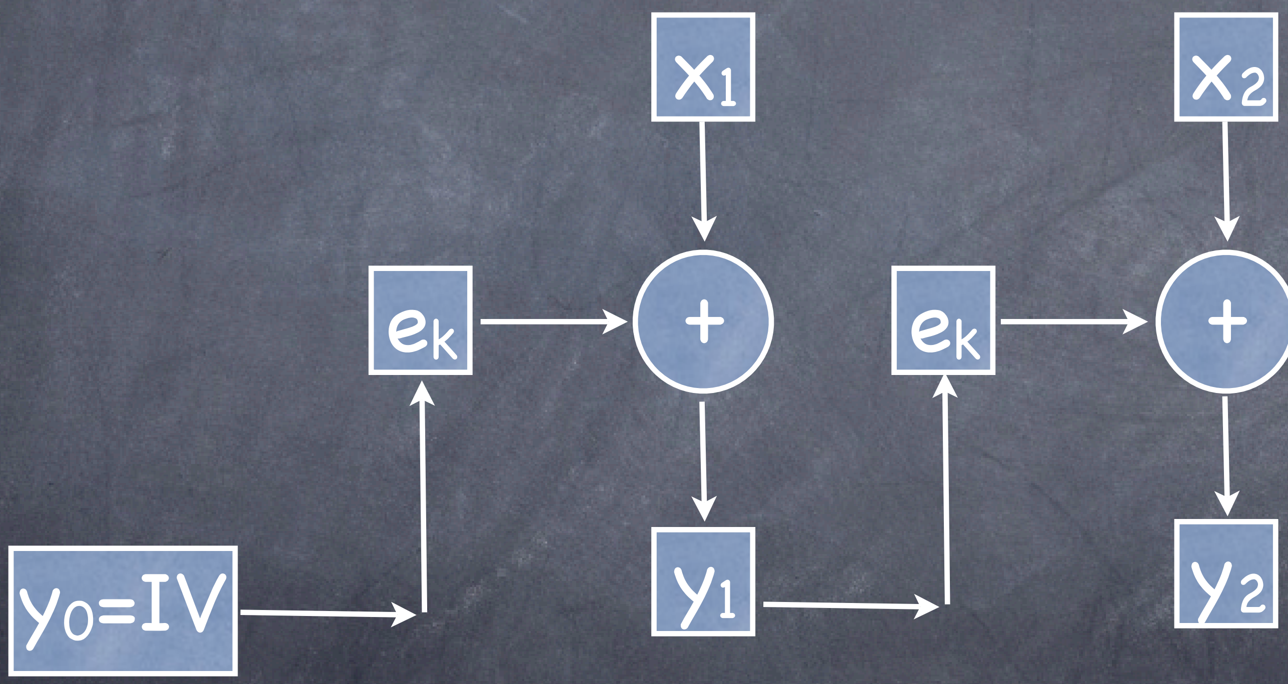
Modes of Operation

- How to use block ciphers when plaintext is more than block length
- ECB (Electronic Codebook Mode):



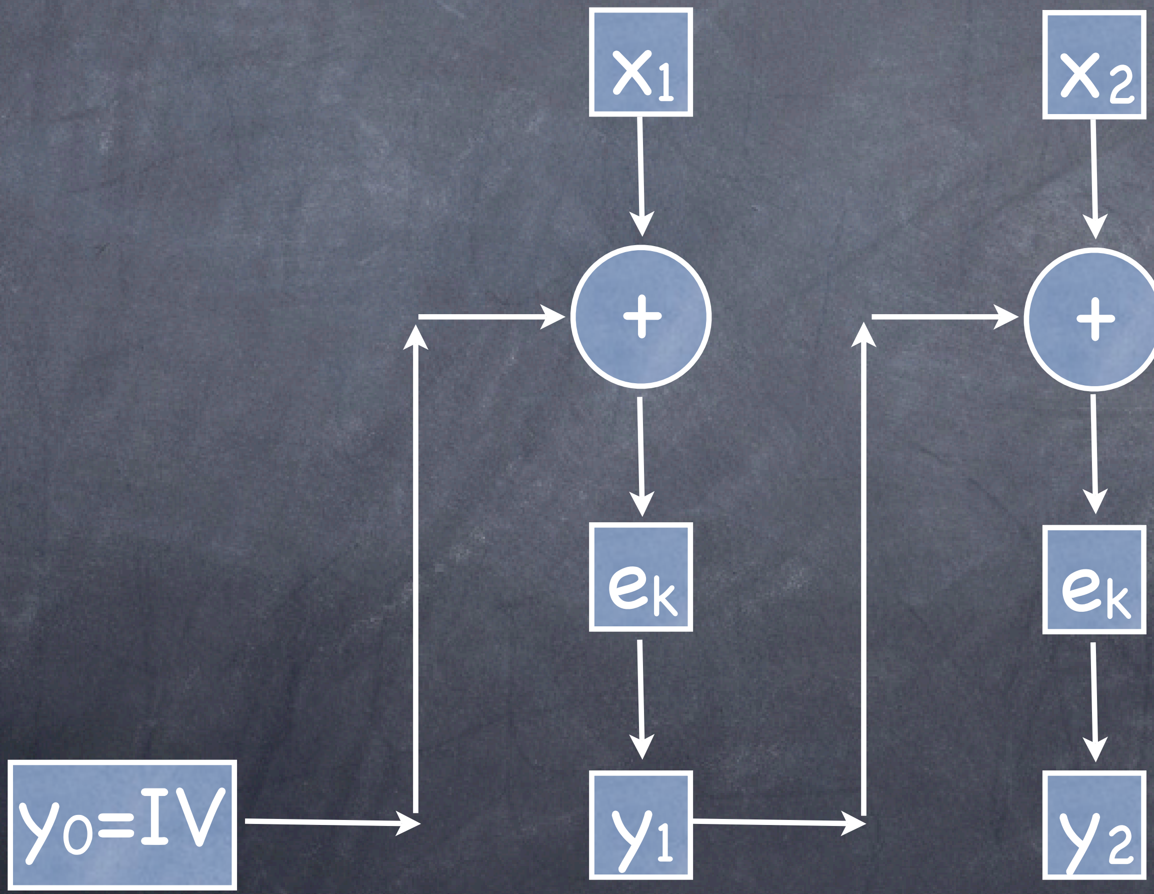
Modes of Operation

- CFB (Cipher Feedback Mode):



Modes of Operation

- CBC (Cipher Block Chaining):



Modes of Operation

- OFB (Output Feedback Mode)