CNT4406/5412 Network Security

Secret Key Cryptography

Zhi Wang

Florida State University

Fall 2013

Introduction

- One key, two operations (encryption/decryption)
- Stream ciphers (e.g., RC4)
  - ➠ take a key and generate a stream of pseudorandom bits (bytes)
  - ➠ XOR pseudorandom bits into data

Introduction

- One key, two operations (encryption/decryption)
- Stream ciphers (e.g., RC4)
  - ➠ take a key and generate a stream of pseudorandom bits (bytes)
  - ➠ XOR pseudorandom bits into data
  - ➠ Shannon proved "XOR with one-time pad" unbreakable
  - ➠ RC4 unbreakable?

Introduction

- One key, two operations (encryption/decryption)
- Stream ciphers (e.g., RC4)
  - ➠ take a key and generate a stream of pseudorandom bits (bytes)
  - ➠ XOR pseudorandom bits into data
  - ➠ Shannon proved "XOR with one-time pad" unbreakable
  - ➠ RC4 unbreakable?
- Block ciphers (e.g., DES, IDEA, AES)
  - ➠ take a key and fixed-size block to generate a fixed-size output
  - ➠ how to encrypt a large messages?
  - ➠ mode of operations: ECB, CBC, CFB, OFB...

Generic Block Encryption

- Block size is usually 64 or 128 bits
  ➠ small block size isn't secure as it is easy to build a table

Generic Block Encryption

- Block size is usually 64 or 128 bits
  ➠ small block size isn't secure as it is easy to build a table
- Input and output has 1:1 mapping (so it can be decrypted)

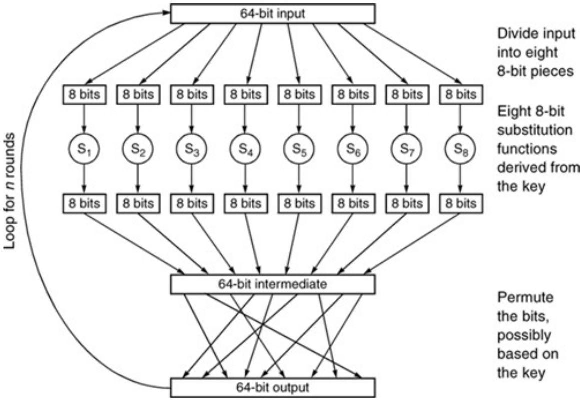
Generic Block Encryption

- Block size is usually 64 or 128 bits
  ➠ small block size isn't secure as it is easy to build a table
- Input and output has 1:1 mapping (so it can be decrypted)
- Output should look random, not correlation to input
  ➠ repeat n rounds to spread the effects of each bit in the input

Generic Block Encryption

- Block size is usually 64 or 128 bits
  ➠ small block size isn't secure as it is easy to build a table
- Input and output has 1:1 mapping (so it can be decrypted)
- Output should look random, not correlation to input
  ➠ repeat n rounds to spread the effects of each bit in the input
- Two operations:
  ➠ substitution: replace one value (8bits) with another (1:1)
  ➠ permutation: shuffle bits around

Generic Block Encryption



DES Overview

- A block cipher, 56-bit key with 8bit parity bits, 64-bit blocks
- Developed at IBM, published in 1977 by NIST
- DES is considered insecure because of its short key
  - in 1998, EFF DES cracker breaks a DES key in 56 hours
  - in 1999, EFF and distributed.net reduced it to 22 hours 15 minutes
  - in 2008, FPGA-based RIVYERA reduced average to $<$ 24 hours

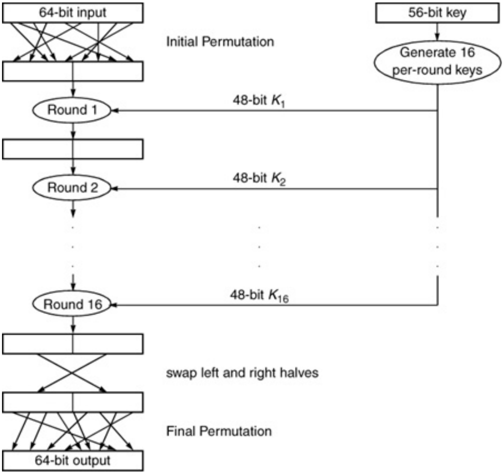
DES Structure

- Encryption
  - initial permutation
  - 16 48-bit per-round keys generated from the 56-bit key
  - 16 DES rounds: 64-bit input + per-round key $\rightarrow$ 64-bit output
  - left and right halves of (64-bit) output swapped
  - final permutation (inverse of the initial permutation)

DES Structure

- Encryption
  - ➠ initial permutation
  - ➠ 16 48-bit per-round keys generated from the 56-bit key
  - ➠ 16 DES rounds: 64-bit input + per-round key → 64-bit output
  - ➠ left and right halves of (64-bit) output swapped
  - ➠ final permutation (inverse of the initial permutation)
- Decryption: running backwards with per-round keys in reverse order

DES Structure



Per-round Keys

- DES key is 56 bits plus 8 parity bits
- Initial permutation to split it into two 28-bit values (C0, D0)
  ➠ no security value
- Per-round keys generated in 16 rounds of rotation and permutation $(K1, K2, ..., K16)$
  ➠ this permutation likely has security value

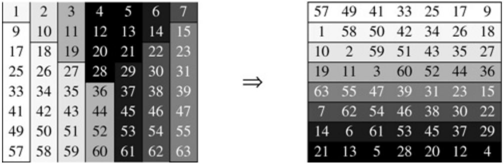
Per-round Keys: Initial Permutation

C_0

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36

D_0

63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4



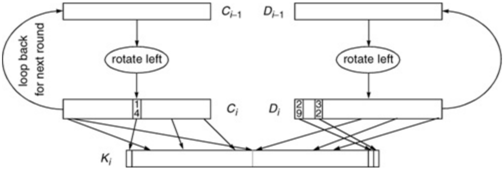
1	2	3	4	5	6	7
9	10	11	12	13	14	15
17	18	19	20	21	22	23
25	26	27	28	29	30	31
33	34	35	36	37	38	39
41	42	43	44	45	46	47
49	50	51	52	53	54	55
57	58	59	60	61	62	63

⇒

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Per-round Keys: 16 Rounds

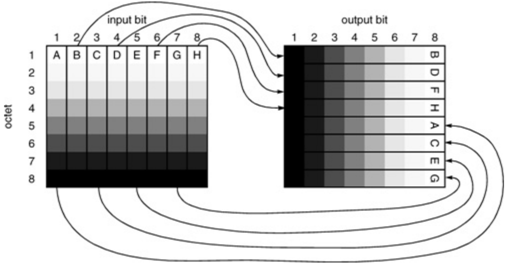
- Rotating left then permutation
  - ➠ a single-bit rotation in rounds 1, 2, 9, 16, two bits in others
  - ➠ 8 Bits are discarded in permutation



14	17	11	24	1	5	41	52	31	37	47	55
3	28	15	6	21	10	30	40	51	45	33	48
23	19	12	4	26	8	44	49	39	56	34	53
16	7	27	20	13	2	46	42	50	36	29	32

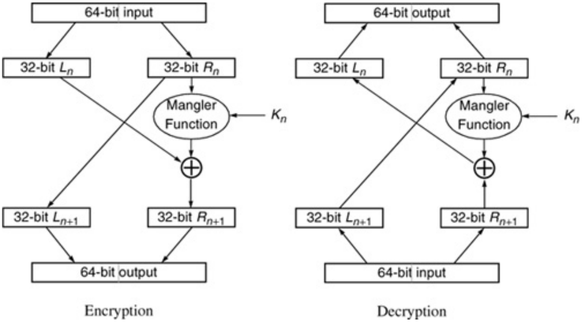
Initial and Final Permutations

- Initial permutation
  - ➠ *i*-th byte into (9-i)th bits
  - ➠ even-numbered bits into byte 1-4, odd-numbered into byte 5-8
- Final permutation is the reverse of initial permutation
- No security value: decrypt innards → decrypt DES



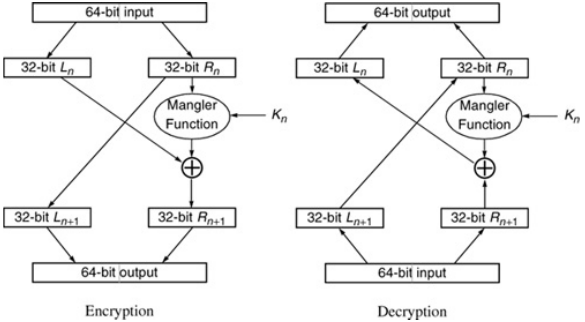
DES Round

- Encryption: $L_{n+1} = R_n, R_{n+1} = L_n \oplus M(R_n, K_n)$



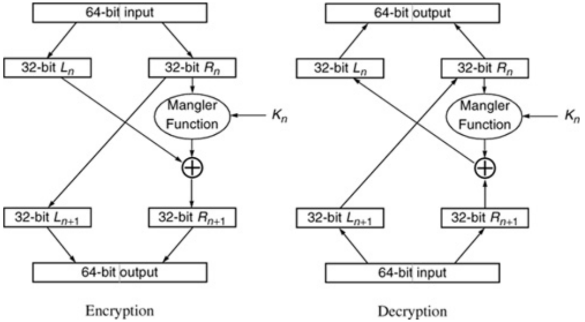
DES Round

- Encryption: $L_{n+1} = R_n, R_{n+1} = L_n \oplus M(R_n, K_n)$
- Decryption: $R_n = L_{n+1}, L_n = R_{n+1} \oplus M(R_n, K_n) = R_{n+1} \oplus M(L_{n+1}, K_n)$
  ➠ mangler function doesn't have to be reversible!



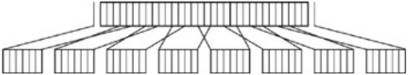
DES Round

- Encryption: $L_{n+1} = R_n, R_{n+1} = L_n \oplus M(R_n, K_n)$
- Decryption: $R_n = L_{n+1}, L_n = R_{n+1} \oplus M(R_n, K_n) = R_{n+1} \oplus M(L_{n+1}, K_n)$
  ➠ mangler function doesn't have to be reversible!
- Two operations are identical with halves swapped



DES Round: Mangler Function

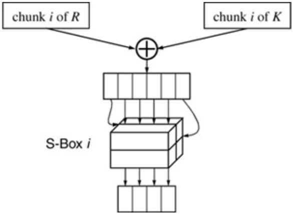
- 32-bit R_n + 48-bit K_n $\rightarrow$ 32-bit output
- Steps:
  - ➠ generate eight 6-bit chunks from R_n



  - ➠ divide 48-bit K_n into eight 6-bit chunks
  - ➠ substitute $Chunk^i_{R_n} \oplus Chunk^i_{K_n}$, (s-box, 6bits $\rightarrow$ 4bits)
  - ➠ combine these 4-bit values into a 32-bit value
  - ➠ permute the 32-bit value to get the output: 16, 7, 20, 21, 29, 12, ..., 30, 6, 22, 11, 4, 25

DES Round: S-Box

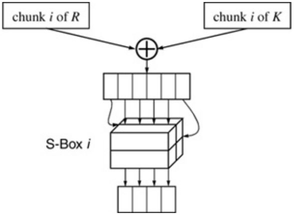
- S-Box: 6-bit input → 4-bit output



DES Round: S-Box

- S-Box: 6-bit input $\rightarrow$ 4-bit output
  - ➠ four 4-bit to 4-bit substitution
  - ➠ input bit 1 and 6 select the substitution to use

Input bits 1 and 6 ↓	Input bits 2 thru 5: 0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	1110	0100	1101	0001	0010	1111	1011	1000	0011	1010	0110	1100	0101	1001	0000	0111
01	0000	1111	0111	0100	1110	0010	1101	0001	1010	0110	1100	1011	1001	0101	0011	1000
10	0100	0001	1110	1000	1101	0110	0010	1011	1111	1100	1001	0111	0011	1010	0101	0000
11	1111	1100	1000	0010	0100	1001	0001	0111	0101	1011	0011	1110	1010	0000	0110	1101



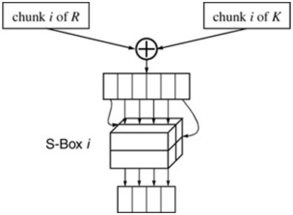
DES Round: S-Box

- S-Box: 6-bit input $\rightarrow$ 4-bit output
  - ➠ four 4-bit to 4-bit substitution
  - ➠ input bit 1 and 6 select the substitution to use

Input bits 1 and 6 / Input bits 2 thru 5

↓	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	1110	0100	1101	0001	0010	1111	1011	1000	0011	1010	0110	1100	0101	1001	0000	0111
01	0000	1111	0111	0100	1110	0010	1101	0001	1010	0110	1100	1011	1001	0101	0011	1000
10	0100	0001	1110	1000	1101	0110	0010	1011	1111	1100	1001	0111	0011	1010	0101	0000
11	1111	1100	1000	0010	0100	1001	0001	0111	0101	1011	0011	1110	1010	0000	0110	1101

- Eight S-Box, one for each chunk



DES Avalanche Effect

- Changes in the plaintext or the key should produce a big change (roughly half of the block size) in the ciphertext
  - $K_1 = 0xCAFEBEEFABEECCCC$
    $K_1\{0\} = 0x0bb7549fc19fbfe0$
    $K_1\{2\} = 0x81ab5276f92eda75$, 27-bit changed
    $K_1\{0xFFFFFFFF\} = 0x25c9cacad0405acd$, 37-bit changed

DES Avalanche Effect

- Changes in the plaintext or the key should produce a big change (roughly half of the block size) in the ciphertext
  - $K_1 = 0xCAFEBEEFABEECCCC$
    $K_1\{0\} = 0x0bb7549fc19fbfe0$
    $K_1\{2\} = 0x81ab5276f92eda75$, 27-bit changed
    $K_1\{0xFFFFFFFF\} = 0x25c9cacad0405acd$, 37-bit changed
  - $K_2 = 0xCAFEBEEFABEECCCE$
    $K_2\{0\} = 0x8c3c710c9e910c9c$, 34-bit changed
    $K_2\{2\} = 0x6521354b2b5bd494$, 35-bit changed
    $K_2\{0xFFFFFFFF\} = 0xd5233ebe3755cab6$, 35-bit changed

DES Weak Keys

- 16 weak DES keys: C_0 and D_0 are all zero, all one, alternating ones and zeros, alternating zeros and ones

DES Weak Keys

- 16 weak DES keys: C_0 and D_0 are all zero, all one, alternating ones and zeros, alternating zeros and ones
  - ➠ four weak keys: C_0 and D_0 are all zero or all one
    weak keys are their own inverse: $K\{K\{m\}\} = m$

DES Weak Keys

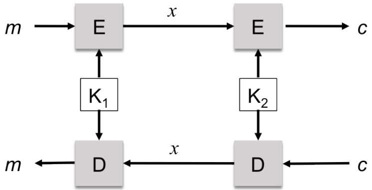
- 16 weak DES keys: C_0 and D_0 are all zero, all one, alternating ones and zeros, alternating zeros and ones
  - ➠ four weak keys: C_0 and D_0 are all zero or all one
    weak keys are their own inverse: $K\{K\{m\}\} = m$
  - ➠ others (12) are semi-weak keys
    each is the inverse of one of the others: $K_1\{K_2\{m\}\} = m$

3DES (Triple DES)

- DES's 56-bit key is too short to be secure
- Can we apply DES multiple times to make it stronger?
- How?

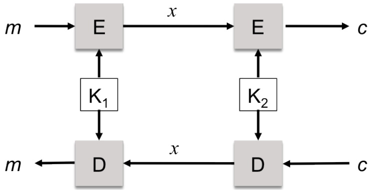
Double Encryption with DES

- Encrypting twice with the same key?



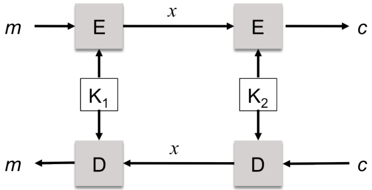
Double Encryption with DES

- Encrypting twice with the same key?
  - ➠ brute-force attack still needs to search only 2^{56} keys



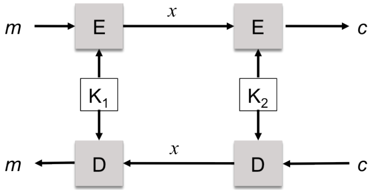
Double Encryption with DES

- Encrypting twice with the same key?
  ➠ brute-force attack still needs to search only 2^{56} keys
- Encrypting twice with two keys?
  ➠ a naive brute-force requires searching 2^{112} keys



Double Encryption with DES

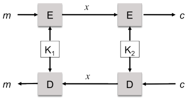
- Encrypting twice with the same key?
  ➠ brute-force attack still needs to search only 2^{56} keys
- Encrypting twice with two keys?
  ➠ a naive brute-force requires searching 2^{112} keys
  ➠ in fact, only need to search about 2^{57} keys: **meet-in-the-middle attack** (a known-plaintext attack)



Meet-in-the-middle Attack

Assume Trudy has a few pairs of <plaintext, ciphertext> encrypted by 2DES:

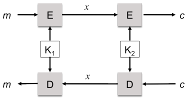
- make table A with 2^{56} entries of $< K_A, E(m_1, K_A) >$ by exhaustively enumerating the DES key K_A, sort it by the second items



Meet-in-the-middle Attack

Assume Trudy has a few pairs of <plaintext, ciphertext> encrypted by 2DES:

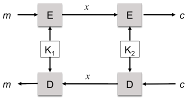
- make table A with 2^{56} entries of $< K_A, E(m_1, K_A) >$ by exhaustively enumerating the DES key K_A, sort it by the second items
- make table B with 2^{56} entries of $< K_B, D(c_1, K_B) >$ by exhaustively enumerating the DES key K_B, sort it by the second item



Meet-in-the-middle Attack

Assume Trudy has a few pairs of <plaintext, ciphertext> encrypted by 2DES:

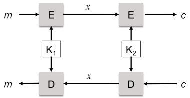
- make table A with 2^{56} entries of $< K_A, E(m_1, K_A) >$ by exhaustively enumerating the DES key K_A, sort it by the second items
- make table B with 2^{56} entries of $< K_B, D(c_1, K_B) >$ by exhaustively enumerating the DES key K_B, sort it by the second item
- search the sorted table with matching entries where $E(m_1, K_A) = D(c_1, K_B)$, such K_A and K_B is a candidate for 2-DES



Meet-in-the-middle Attack

Assume Trudy has a few pairs of <plaintext, ciphertext> encrypted by 2DES:

- make table A with 2^{56} entries of $< K_A, E(m_1, K_A) >$ by exhaustively enumerating the DES key K_A, sort it by the second items
- make table B with 2^{56} entries of $< K_B, D(c_1, K_B) >$ by exhaustively enumerating the DES key K_B, sort it by the second item
- search the sorted table with matching entries where $E(m_1, K_A) = D(c_1, K_B)$, such K_A and K_B is a candidate for 2-DES
- test the candidates with $< m_2, c_2 >$, then $< m_3, c_3 >$, ..., only the correct key pair will work for all of them

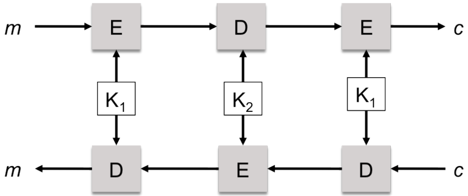


Meet-in-the-middle Attack...

- It converges quickly when testing candidates against $< m_i, c_i >$
  - ➠ each table contains 2^{56} blocks out of 2^{64} possible blocks
  - ➠ each block has $\frac{1}{256}$ chance of appearing in a table
  - ➠ about 2^{48} entries of Table A also appear in Table B
  - ➠ if $< K_A, K_B >$ is an imposer, the chance of $E(m_2, K_A) = D(c_2, K_B)$ is about $\frac{1}{2^{16}}$
  - ➠ each testing of $< m_i, c_i >$ reduces the chance by a factor of $\frac{1}{2^{64}}$
- Computation complexity: $O(2^{56})$ assuming enough space is provided to sort table A and B in $O(2^{56})$

3DES

- 2 keys used instead of 3 keys
  ➠ equivalent key length is 112 bits
- 3DES operations: EDE for encryption, DED for decryption
  ➠ 3DES is inefficient and expensive



IDEA Overview

- Published in 1991 by ETH Zurich
- Structurally similar to DES, 64-bit blocks and 128-bit key
- IDEA is relatively slow

Primitive Operations

Three reversible operations on 16-bit quantities

- $\oplus$

Primitive Operations

Three reversible operations on 16-bit quantities

- $\oplus$
- $+ \mod 2^{16}$

Primitive Operations

Three reversible operations on 16-bit quantities

- $\oplus$
- $+ \mod 2^{16}$
- $\times \mod (2^{16}+1)$
  - for each q, there is a p having $pq = 1 \mod (2^{16}+1)$

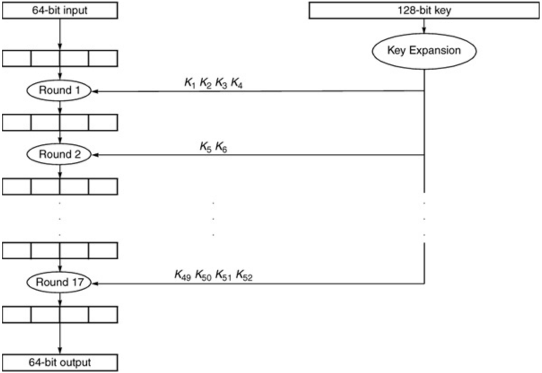
Primitive Operations

Three reversible operations on 16-bit quantities

- $\oplus$
- $+ \mod 2^{16}$
- $\times \mod (2^{16}+1)$
  - for each q, there is a p having $pq = 1 \mod (2^{16}+1)$
  - Euclid's algorithm: $gcd(x, y) = nx + vy$
    let $y = 2^{16} + 1$ and $x = q$
    because y is a prime and $q < y$, $gcd(q, y) = 1$
    using Euclid's algorithm, we can get n that $1 = nq + v(2^{16} + 1)$

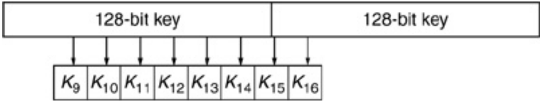
IDEA Structure

- Expand 128-bit key into 52 16-bit keys
- 17 rounds, odd rounds use 4 keys, even rounds use 2 keys
- All operations on 16-bit quantities



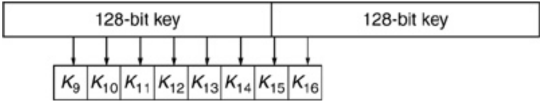
Key Expansion

- Encryption: repeat the following steps 6 times
  - left rotate the 128-bit key by $25 \times i \mod 128$ times
  - output 8 16-bit keys from the key
  - output last 4 16-bit keys starting at bit 23



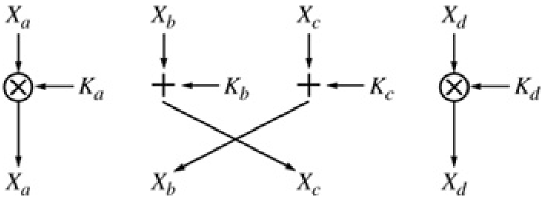
Key Expansion

- Encryption: repeat the following steps 6 times
  - ➠ left rotate the 128-bit key by $25 \times i \mod 128$ times
  - ➠ output 8 16-bit keys from the key
  - ➠ output last 4 16-bit keys starting at bit 23
- Decryption
  - ➠ generate the same keys, but use them backwards
  - ➠ inverse the odd-round keys, but keep the even-round keys



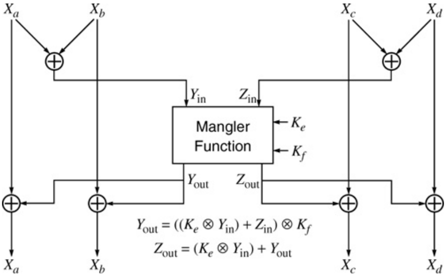
Odd Rounds

- Treat 64-bit input as 4 16-bit quantities (X_a, X_b, X_c, X_d)
- Use 4 16-bit keys (K_a, K_b, K_c, K_d)
- $X'_a = X_a \times K_a, X'_b = X_c + K_c, X'_c = X_b + K_b, X'_d = X_d K_d$
- Use the inverse of the keys to reverse the round



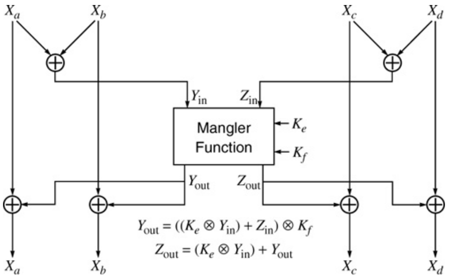
Even Rounds

- Encryption
  $Y_{in} = X_a \oplus X_b, Z_{in} = X_c \oplus X_d, Y_{out} = ((K_e \times Y_{in}) + Z_{in}) \times K_f, Z_{out} = (K_e \times Y_{in}) + Y_{out}$
  $X'_a = X_a \oplus Y_{out}, X'_b = X_b \oplus Y_{out}, X'_c = X_c \oplus Z_{out}, X'_d = X_d \oplus Z_{out}$



Even Rounds

- Encryption
  $Y_{in} = X_a \oplus X_b, Z_{in} = X_c \oplus X_d, Y_{out} = ((K_e \times Y_{in}) + Z_{in}) \times K_f, Z_{out} = (K_e \times Y_{in}) + Y_{out}$
  $X_a' = X_a \oplus Y_{out}, X_b' = X_b \oplus Y_{out}, X_c' = X_c \oplus Z_{out}, X_d' = X_d \oplus Z_{out}$
- Decryption: just feed it with $X_a', \ldots, X_d'$
  $X_a' \oplus X_b' = Y_{in}, X_c' \oplus X_d' = Z_{in} \rightarrow X_a = X_a' \oplus Y_{out}, X_b = X_b' \oplus Y_{out} \ldots$
  $\rightarrow$: inputs to the mangler function are the same!



Encrypting Large Messages

- Most secret key ciphers are block cipher w/ fixed size input
- How to encrypt a large message?

Encrypting Large Messages

- Most secret key ciphers are block cipher w/ fixed size input
- How to encrypt a large message?
  - ➠ Electronic Code Book (ECB)
  - ➠ Cipher Block Chaining (CBC)
  - ➠ k-Bit Cipher Feedback Mode (CFB)
  - ➠ k-Bit Output Feedback Mode (OFB)
  - ➠ Counter Mode (CTR)

Issues to Consider

- Information leakage
  ➠ does it reveal info about the plaintext blocks?

Issues to Consider

- Information leakage
  ➠ does it reveal info about the plaintext blocks?
- Ciphertext manipulation
  ➠ can an attacker modify ciphertext in a way that will produce a predictable/desired change in the decrypted plaintext?

Issues to Consider

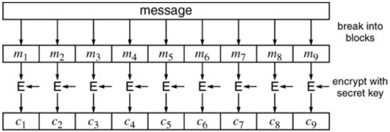
- Information leakage
  ➠ does it reveal info about the plaintext blocks?
- Ciphertext manipulation
  ➠ can an attacker modify ciphertext in a way that will produce a predictable/desired change in the decrypted plaintext?
- Parallel/Sequential
  ➠ can the cipher encrypt/decrypt blocks in parallel?

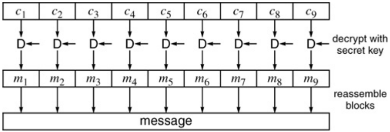
Issues to Consider

- Information leakage
  ➠ does it reveal info about the plaintext blocks?
- Ciphertext manipulation
  ➠ can an attacker modify ciphertext in a way that will produce a predictable/desired change in the decrypted plaintext?
- Parallel/Sequential
  ➠ can the cipher encrypt/decrypt blocks in parallel?
- Error propagation
  ➠ how many blocks will be affected by a garbled ciphertext block?

Electronic Code Book (ECB)

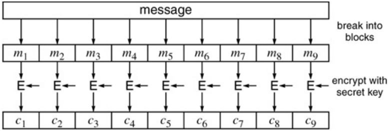
- Encrypt each block independently with the key, decrypt the same

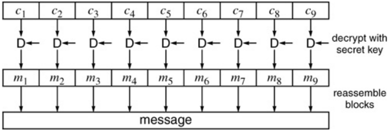




Electronic Code Book (ECB)

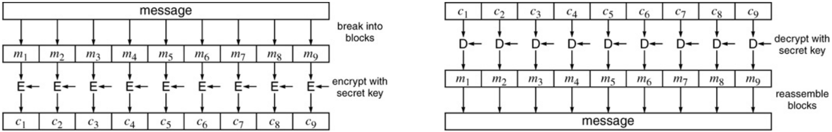
- Encrypt each block independently with the key, decrypt the same
- Information leakage?





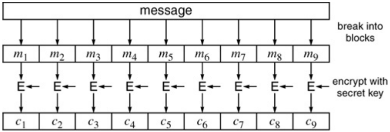
Electronic Code Book (ECB)

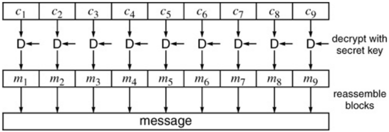
- Encrypt each block independently with the key, decrypt the same
- Information leakage?
  - ➠ ciphertext for identical plaintext blocks are the same



Electronic Code Book (ECB)

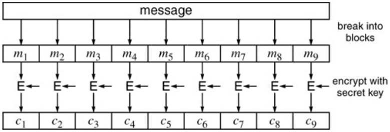
- Encrypt each block independently with the key, decrypt the same
- Information leakage?
  - ➠ ciphertext for identical plaintext blocks are the same
- Ciphertext manipulation?

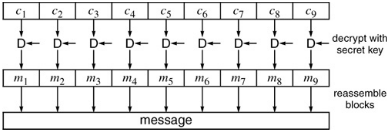




Electronic Code Book (ECB)

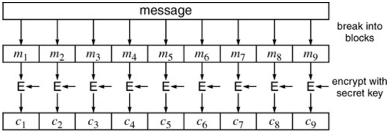
- Encrypt each block independently with the key, decrypt the same
- Information leakage?
  ➠ ciphertext for identical plaintext blocks are the same
- Ciphertext manipulation?
  ➠ attacker can cut and paste ciphertext blocks

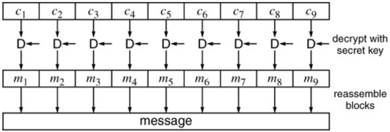




Electronic Code Book (ECB)

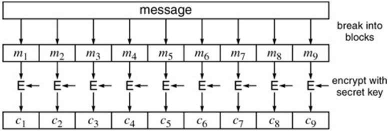
- Encrypt each block independently with the key, decrypt the same
- Information leakage?
  ➠ ciphertext for identical plaintext blocks are the same
- Ciphertext manipulation?
  ➠ attacker can cut and paste ciphertext blocks
- Parallel encryption/decryption?

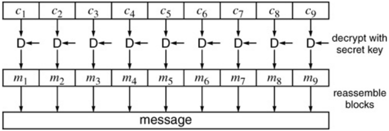




Electronic Code Book (ECB)

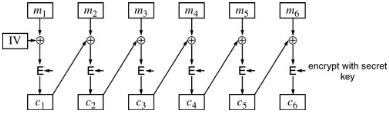
- Encrypt each block independently with the key, decrypt the same
- Information leakage?
  ➠ ciphertext for identical plaintext blocks are the same
- Ciphertext manipulation?
  ➠ attacker can cut and paste ciphertext blocks
- Parallel encryption/decryption?
- Error propagation?

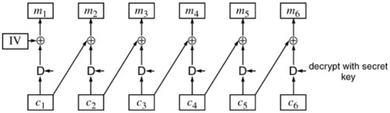




Cipher Block Chaining (CBC)

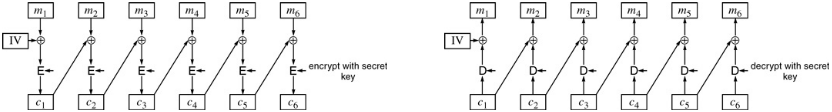
- Encryption
  - ➠ $\oplus$ previous ciphertext block to the message block, then encrypt it
  - ➠ $C_n = K\{m_n \oplus C_{n-1}\} = E(m_n \oplus C_{n-1}, K)$
  - ➠ use IV (not secret) so ciphertext of same messages is different
  - ➠ each ciphertext block depends on all previous blocks





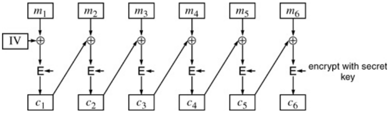
Cipher Block Chaining (CBC)

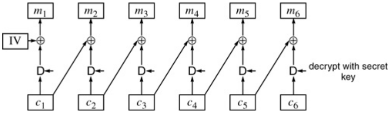
- Encryption
  - ➠ $\oplus$ previous ciphertext block to the message block, then encrypt it
  - ➠ $C_n = K\{m_n \oplus C_{n-1}\} = E(m_n \oplus C_{n-1}, K)$
  - ➠ use IV (not secret) so ciphertext of same messages is different
  - ➠ each ciphertext block depends on all previous blocks
- Decryption
  - ➠ $m_n = D(C_n, K) \oplus C_{n-1}$
  - ➠ each plaintext block depends on ??? ciphertext blocks



Cipher Block Chaining (CBC)

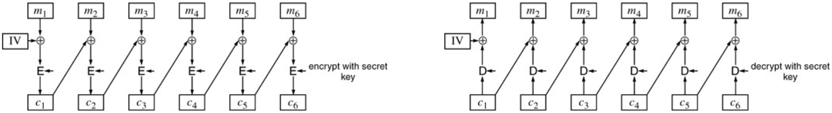
- Information leakage?





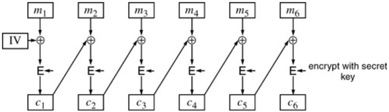
Cipher Block Chaining (CBC)

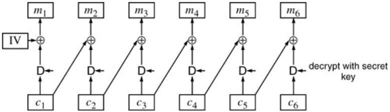
- Information leakage?
  - ➠ ciphertext for identical plaintext blocks are different



Cipher Block Chaining (CBC)

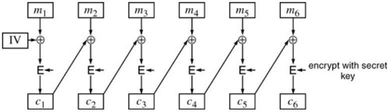
- Information leakage?
  ➠ ciphertext for identical plaintext blocks are different
- Ciphertext manipulation?

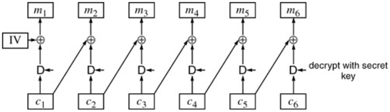




Cipher Block Chaining (CBC)

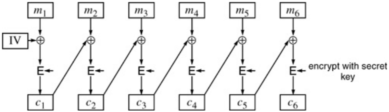
- Information leakage?
  ➠ ciphertext for identical plaintext blocks are different
- Ciphertext manipulation?
  ➠ modifying c_n predictably changes m_{n+1} ($= D(c_{n+1}, K) \oplus c_n$), but garbles m_n because $m_n = D(c_n, K) \oplus c_{n-1}$

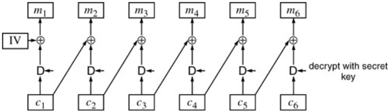




Cipher Block Chaining (CBC)

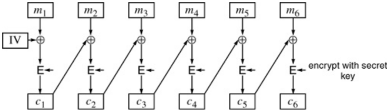
- Information leakage?
  ➠ ciphertext for identical plaintext blocks are different
- Ciphertext manipulation?
  ➠ modifying c_n predictably changes m_{n+1} $(= D(c_{n+1}, K) \oplus c_n)$, but garbles m_n because $m_n = D(c_n, K) \oplus c_{n-1}$
  ➠ rearranging ciphertext blocks with known $< m_i, c_i >$ pairs allows calculation of decrypted plaintext: $m'_n = D(c'_n, K) \oplus c_{n-1}$

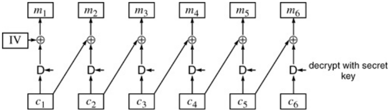




Cipher Block Chaining (CBC)

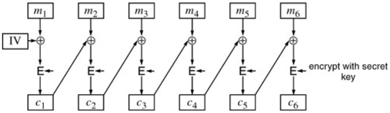
- Information leakage?
  ➠ ciphertext for identical plaintext blocks are different
- Ciphertext manipulation?
  ➠ modifying c_n predictably changes m_{n+1} $(= D(c_{n+1}, K) \oplus c_n)$, but garbles m_n because $m_n = D(c_n, K) \oplus c_{n-1}$
  ➠ rearranging ciphertext blocks with known $< m_i, c_i >$ pairs allows calculation of decrypted plaintext: $m'_n = D(c'_n, K) \oplus c_{n-1}$
- Parallel encryption/decryption?

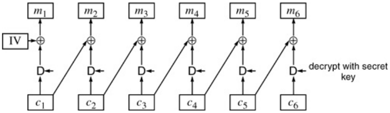




Cipher Block Chaining (CBC)

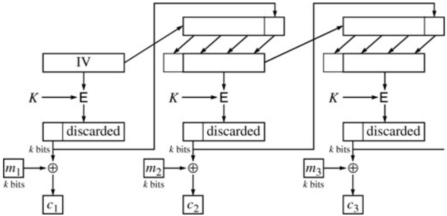
- Information leakage?
  ➠ ciphertext for identical plaintext blocks are different
- Ciphertext manipulation?
  ➠ modifying c_n predictably changes m_{n+1} $(= D(c_{n+1}, K) \oplus c_n)$, but garbles m_n because $m_n = D(c_n, K) \oplus c_{n-1}$
  ➠ rearranging ciphertext blocks with known $< m_i, c_i >$ pairs allows calculation of decrypted plaintext: $m'_n = D(c'_n, K) \oplus c_{n-1}$
- Parallel encryption/decryption?
- Error propagation?





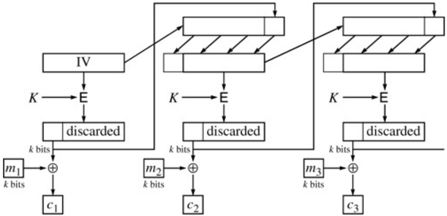
Output Feedback Mode (OFB)

- OFB is a stream cipher:one-time pad to be $\oplus$'ed to message
- 64-bit OFB has a one-time pad of $b_0|b_1|b_2|b_3|\ldots$ with $b_0 = K\{IV\}, b_1 = K\{b_0\}, b_2 = K\{b_1\}\ldots$



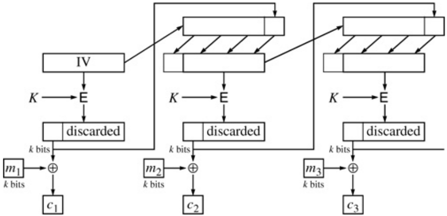
Output Feedback Mode (OFB)

- OFB is a stream cipher:one-time pad to be $\oplus$'ed to message
- 64-bit OFB has a one-time pad of $b_0|b_1|b_2|b_3|$... with $b_0 = K\{IV\}, b_1 = K\{b_0\}, b_2 = K\{b_1\}$...
  ➠ IV must never repeat!!!



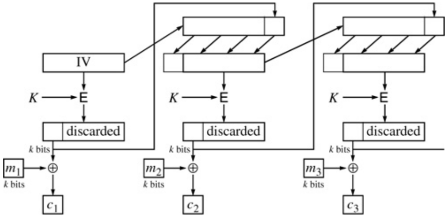
Output Feedback Mode (OFB)

- OFB is a stream cipher:one-time pad to be $\oplus$'ed to message
- 64-bit OFB has a one-time pad of $b_0|b_1|b_2|b_3|\ldots$ with $b_0 = K\{IV\}, b_1 = K\{b_0\}, b_2 = K\{b_1\}\ldots$
  - ➠ IV must never repeat!!!
  - ➠ pad is independent of the message, can be generated in advance



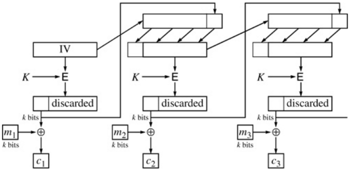
Output Feedback Mode (OFB)

- OFB is a stream cipher:one-time pad to be $\oplus$'ed to message
- 64-bit OFB has a one-time pad of $b_0|b_1|b_2|b_3|$... with $b_0 = K\{IV\}, b_1 = K\{b_0\}, b_2 = K\{b_1\}$...
  - ➠ IV must never repeat!!!
  - ➠ pad is independent of the message, can be generated in advance
  - ➠ k-bit OFB: only k bits of b_n are used



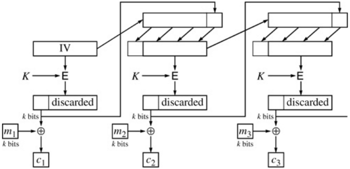
Output Feedback Mode (OFB)

- Information leakage?



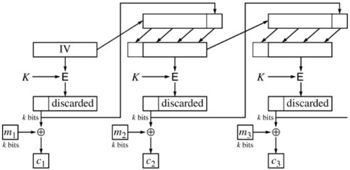
Output Feedback Mode (OFB)

- Information leakage?
  - ➠ ciphertext for identical plaintext blocks are different



Output Feedback Mode (OFB)

- Information leakage?
  ➠ ciphertext for identical plaintext blocks are different
- Ciphertext manipulation?

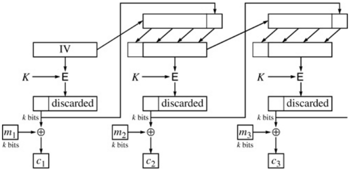


Output Feedback Mode (OFB)

- Information leakage?

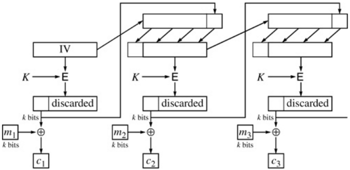
  ➠ ciphertext for identical plaintext blocks are different
- Ciphertext manipulation?

  ➠ with known $< m_i, c_i >$, the attacker can set decrypted plaintext to any value by replacing c_i with $c_i \oplus m_i \oplus m_i'$



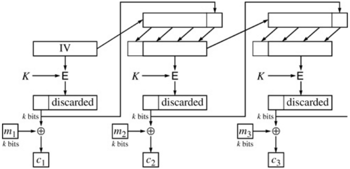
Output Feedback Mode (OFB)

- Information leakage?
  ➠ ciphertext for identical plaintext blocks are different
- Ciphertext manipulation?
  ➠ with known $< m_i, c_i >$, the attacker can set decrypted plaintext to any value by replacing c_i with $c_i \oplus m_i \oplus m_i'$
- Parallel encryption/decryption?



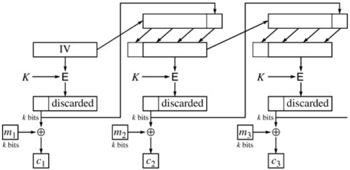
Output Feedback Mode (OFB)

- Information leakage?
  ➠ ciphertext for identical plaintext blocks are different
- Ciphertext manipulation?
  ➠ with known $< m_i, c_i >$, the attacker can set decrypted plaintext to any value by replacing c_i with $c_i \oplus m_i \oplus m'_i$
- Parallel encryption/decryption?
  ➠ one-time pad can be generated sequentially
  ➠ parallel encryption/decryption is possible with pre-generated pad



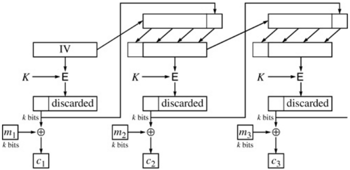
Output Feedback Mode (OFB)

- Information leakage?
  ➠ ciphertext for identical plaintext blocks are different
- Ciphertext manipulation?
  ➠ with known $< m_i, c_i >$, the attacker can set decrypted plaintext to any value by replacing c_i with $c_i \oplus m_i \oplus m_i'$
- Parallel encryption/decryption?
  ➠ one-time pad can be generated sequentially
  ➠ parallel encryption/decryption is possible with pre-generated pad
- Error propagation?



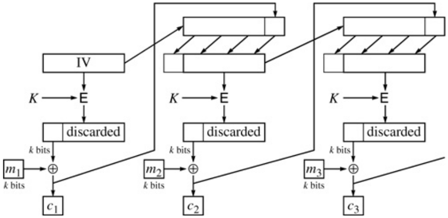
Output Feedback Mode (OFB)

- Information leakage?
  ➠ ciphertext for identical plaintext blocks are different
- Ciphertext manipulation?
  ➠ with known $< m_i, c_i >$, the attacker can set decrypted plaintext to any value by replacing c_i with $c_i \oplus m_i \oplus m'_i$
- Parallel encryption/decryption?
  ➠ one-time pad can be generated sequentially
  ➠ parallel encryption/decryption is possible with pre-generated pad
- Error propagation?
  ➠ only bits corresponding to the garbled bits in ciphertext



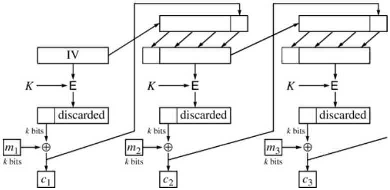
Cipher Feedback Mode (CFB)

- CFB is a stream cipher very similar to OFB
  - k bits shifted are k-bit ciphertext, instead of k-bit one-time pad
  - one-time pad cannot be generated in advance
  - IV is less critical



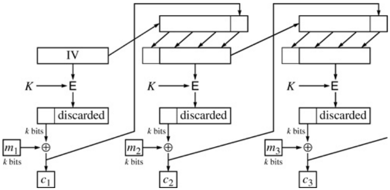
Cipher Feedback Mode (CFB)

- Information leakage?
  - ➠ ciphertext for identical plaintext blocks are different



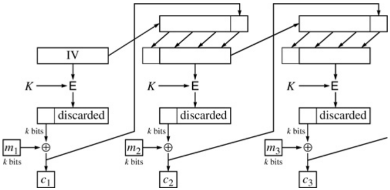
Cipher Feedback Mode (CFB)

- Information leakage?
  ➠ ciphertext for identical plaintext blocks are different
- Ciphertext manipulation?
  ➠ modifying c_n predictably changes m_n, but garbles ??? blocks



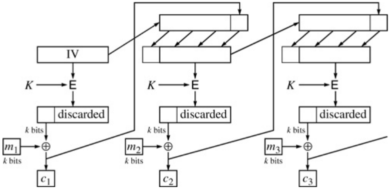
Cipher Feedback Mode (CFB)

- Information leakage?
  - ➠ ciphertext for identical plaintext blocks are different
- Ciphertext manipulation?
  - ➠ modifying c_n predictably changes m_n, but garbles ??? blocks
- Parallel encryption/decryption?
  - ➠ encyption - No, decryption - Yes, why???



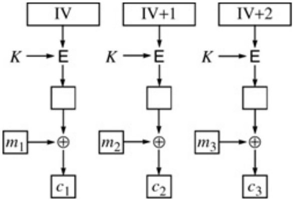
Cipher Feedback Mode (CFB)

- Information leakage?
  ➠ ciphertext for identical plaintext blocks are different
- Ciphertext manipulation?
  ➠ modifying c_n predictably changes m_n, but garbles ??? blocks
- Parallel encryption/decryption?
  ➠ encyption - No, decryption - Yes, why???
- Error propagation?
  ➠ resynchronize decryption against n k-bits insertion or deletion



Counter Mode

- Stream cipher: one-time pad is $K\{IV\}, ..., K\{IV+n\}, ...$
  - ➠ one-time pad can be pre-computed
  - ➠ parallel encryption/decryption is supported
  - ➠ IV must never repeat!!!

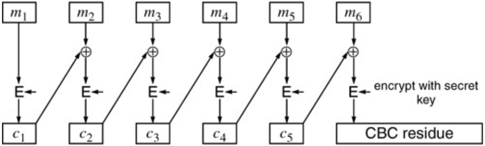


Message Authentication

- Encryption provides confidentiality for a message
- How to use encryption to authenticate a message?
  - ➠ prove the message was created by someone with the key
  - ➠ prove it hasn't been modified except by someone with the key

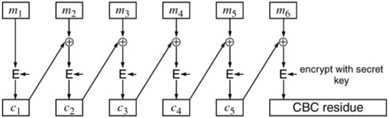
Message Integrity with CBC Residue

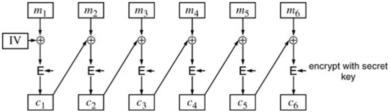
- Encrypt message using CBC mode with IV set to 0
- The final ciphertext block is called CBC residue, transmit it with the plaintext
  - ➠ CBC residue depends on all previous blocks
  - ➠ only someone with the key can generate the correct CBC residue (except with a probability of $\frac{1}{2^{64}}$)



Ensuring Privacy and Integrity

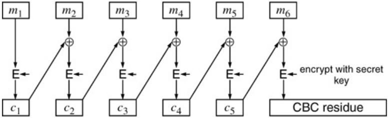
- Encryption alone doesn't guarantee integrity
  - ➠ decryption just transfers the ciphertext to some message

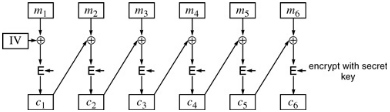




Ensuring Privacy and Integrity

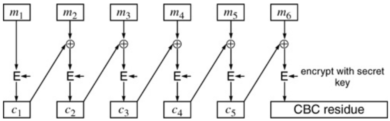
- Encryption alone doesn't guarantee integrity
  ➠ decryption just transfers the ciphertext to some message
- CBC encryption plus CBC residue doesn't work
  ➠ just repeat the final ciphertext block as the CBC residue

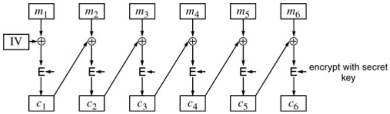




Ensuring Privacy and Integrity

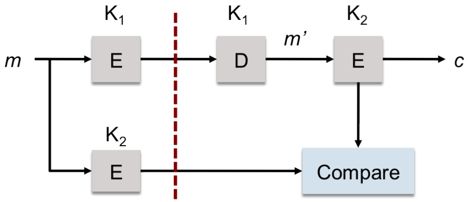
- Encryption alone doesn't guarantee integrity
  ➠ decryption just transfers the ciphertext to some message
- CBC encryption plus CBC residue doesn't work
  ➠ just repeat the final ciphertext block as the CBC residue
- CBC encryption of plaintext plus CBC residue doesn't work
  ➠ the final ciphertext block is always K0, why?





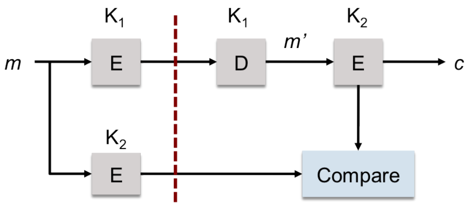
Ensuring Privacy and Integrity

- Encrypt message with K_1, compute CBC residue with K_2
  ➠ guarantee privacy and integrity, expensively



Ensuring Privacy and Integrity

- Encrypt message with K_1, compute CBC residue with K_2
  - ➠ guarantee privacy and integrity, expensively
- Variations of the scheme
  - ➠ transform K_1 into K_2 (K_1 and K_2 are related)
  - ➠ use weak (cheaper) cryptographic checksum
  - ➠ use cryptographic hash instead of CBC residue



Summary

- Stream cipher and block cipher
- DES, 3DES, and IDEA
- Modes of operation: ECB, CBC, CFB, OFB
- Message integrity

- Next class: cryptographic hash function