Block Ciphers

Gianluca Dini
Dept. of Ingegneria dell'Informazione
University of Pisa
gianluca.dini@unipi.it
Version: 2024-03-11

1

Block Ciphers

GENERAL CONCEPTS

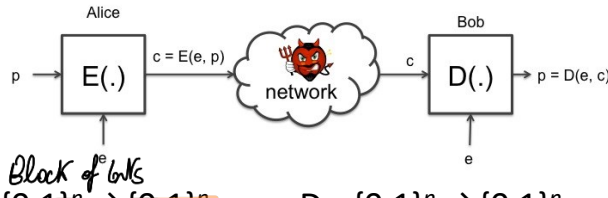
2

OTP and stream ciphers encrypt bit wise

Block cipher

Comm. scheme is always the same.

- Block ciphers break up the plaintext in blocks of fixed length *n* bits and encrypt one block at time



Block of n bits

- E_k: $\{0,1\}^n \rightarrow \{0,1\}^n$ D_k: $\{0,1\}^n \rightarrow \{0,1\}^n$
- E is a keyed permutation: $E(k, p) = E_k(p) = Enc_k(p)$
- $E_K(\cdot)$ is a permutation

• We take a block of bits and encrypt it. ①

3

① Instead of 1 bit at a time, we encrypt in block. When I fix the key we have a function that takes n bits and outputs n bits.

Permutation

- E_k is a permutation
  - E_K is efficiently computable
  - E_k is bijective (invertible)
    - Surjective (or onto)
    - Injective (or one-to-one)
  - E_k^{-1} is efficiently computable

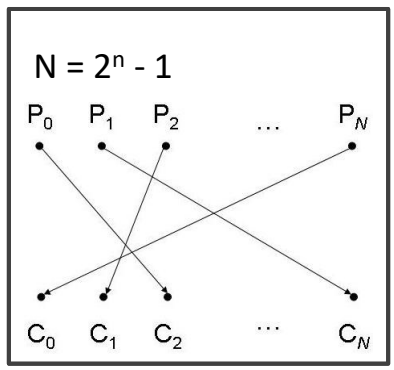
4

Examples

- Block ciphers
  - DES n = 64 bits, k = 56 bits
  - 3DES n = 64 bits, k = 168 bits
  - AES n = 128 bits k = 128, 192, 256 bits

5

Random permutations



A possible random permutation π

- Let $Perm_n$ be the set of all permutations $\pi: \{0,1\}^n \rightarrow \{0,1\}^n$
- $|Perm_n| = 2^n!$
- A true random cipher
  - implements all the permutations in $Perm_n$
  - uniformly selects a permutation $\pi \in Perm_n$ at random

fixing a key means chosing a permutation.

7

True Random Cipher

- A True random cipher is perfect
- A true random cipher implements all possible Random permutations ($2^n!$) → need a key to give a name to all
  - Need a uniform random key for each permutation of the perms. (naming)
    - key size := $\log_2(2^n!) \approx (n - 1.44)\, 2^n$ APPROXIMATION
      - Exponential in the block size! Size of key grows exp. to the block size
      - The block size cannot be small to avoid a dictionary attack
- A true random cipher cannot be implemented

① Why not a small n? n=4, so $2^4 \cdot 4 \tilde{=} 64$ bits. Here you get exposed to dictionary attack.

Mar-25 Block Ciphers 8

8

C	P
0000	0110
⋮	⋮

Adversary collects cyph and corresponding plaintexts. If n=4, each entry is 4 bits and 16 combinations. So you can translate without a key.

Pseudorandom permutations

- Consider a *family of permutations* parametrized by $\kappa \in K = \{0,1\}^k$, $E_\kappa: \{0,1\}^n \to \{0,1\}^n$ ↑ we implement only a subset, that must be large enough
- A E_κ is a *pseudorandom permutation* (PRP) if it is indistinguishable from a uniform random permutation by a limited adversary
- $|\{E_\kappa\}| = 2^k << |\text{Perm}_n|$, with $|\kappa| = k$
- A block cipher is a practical instantiation of a PRP

Key of K bits: I can implement the 2^K perm instead of all of them

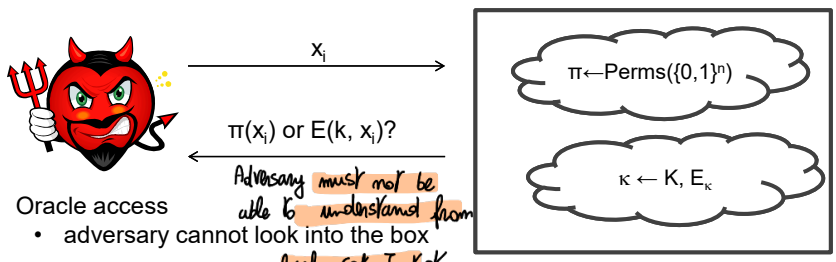
Mar-25 Block Ciphers 9

9

We want no efficient algorithm for a limited adversary to tell the difference between using all permutations or a subset of it.

Practical block cipher

- In practice, the encryption function corresponding to a randomly chosen key should appear as a randomly chosen permutation to a limited adversary



- Oracle access
  - adversary cannot look into the box

Adversary must not be able to understand from which set I took the permutation. This is the notion of security.

Mar-25 Block Ciphers 10

10

Exhaustive key search attack

↳ Basically a known plaintext attack

- The attack
  - Given a pair (pt, ct), check whether ct == E_{ki}(pt), i = 0, 1, …, $2^k - 1$
    - Known-plaintext attack
    - Time complexity: $O(2^k)$
- False positives
  - Do you expect that just one key k maps pt into ct? ①
  - How many keys (false positives) do we expect to map pt into ct?
  - How do you discriminate the good one?

① For OTP, set of keys is minimal: there exist only 1 key that encrypts a given plaintext in a given ciphertext

mar-25 Block Ciphers

11

PANKO REMINDS:

there where exist several keys that encrypt a given plaintext to a certain ciphertext

Exhaustive key search

- False positives
  - Do you expect that just one key k maps pt into ct?
  - How many keys (false positives) do we expect to map pt into ct?
  - How do you discriminate the good one?

NOTE: how do we get rid of false positives? If I have multiple pairs I have two sets of keys that map pt1 to ct1 and pt2 to ct2. If I intersect those two I might have just a set containing the right key, or multiple ones. In that case I need multiple pairs.

12

False positives

- Problem: Given (ct, pt) s.t. $ct = E_{k^*}(pt)$ for a given k^*, determine the number of keys that map pt into ct

the key we want

- Solution.
  - Given a certain key k, $P(k) = Pr[E_k(pt) == ct] = 1/2^n$
  - The *expected* number of keys that map pt into ct is $2^k \times 1/2^n = 2^{k-n}$

2^n is the # of all possible blocks

# of all possible keys times the probability to have a match

It is expected

13

False positives

- Example 1 – DES with n = 64 and k = 56
  - On average 2^{-8} keys map pt into ct
  - One pair (pt, ct) is sufficient for an exhaustive key search
- Example 2 – Skipjack with n = 64 and k = 80
  - On average 2^{16} keys map pt into ct
  - Two or more plaintext-ciphertext pairs are necessary for an exhaustive key search

Mar-25 Block Ciphers 14

14

False positives

- Consider now t pairs (pt_i, ct_i), i = 1, 2,…, t
  - Given k, $Pr[E_k(pt_i) = ct_i$, for all i = 1, 2,…, t] = $1/2^{tn}$
  - Expected number of keys that map pt_i into ct_i, for all i = 1, 2, …, t, is $2^k/2^{tn} = 2^{k-tn}$
- Example 3 – Skypjack with k = 80, n = 64, t = 2
  - The expected number of keys is = $2^{80-2\times64} = 2^{-48}$
  - Two pairs are sufficient for an exhaustive key search

Mar-25 Block Ciphers 15

15

False positives

false: $2^{k-tn}-1$?

- THEOREM
  - Given a block cipher with a key lenght of *k* bits and a block size of *n* bits, as well as *t* plaintext-ciphertext pairs, (pt_1, ct_1),…, (pt_t, ct_t), the expected number of false keys which encrypt all plaintexts to the corresponding ciphertexts is 2^{k-tn}
- FACT
  - Two input-output pairs are generally enough for exhaustive key search

Mar-25 Block Ciphers 16

16

Block ciphers

EXERCISES

Mar-25 Block Ciphers 17

17

Exercise 1 - Exhaustive key search

- Exhaustive key search is a known-plaintext attack
- However, the adversary can mount a cyphertext-only attack if (s)he has some knowledge on PT

18

Exercise 1 – exhaustive key search

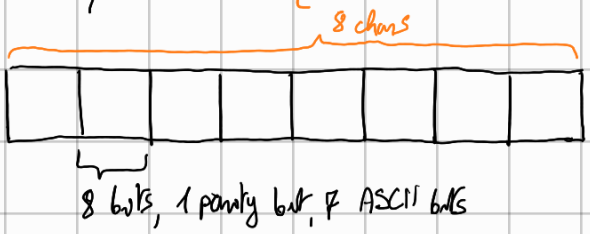
- Assume DES is used to encrypt 64-bit blocks of 8 ASCII chars, with one bit per char serving as parity bit
- How many CT blocks the adversary needs to remove false positives with a probability smaller than ε?

19

Exhaustive key search is a known plaintext attack, but it can also be a ciphertext only attack providing we have information about plaintext.

Take a look at DES: Very good algorithm, there are no good cryptoanalysis mechanisms known.

DES ($n=64$, $k=56$) [A block contains 8 chars]



Idea: suppose I intercept a block and start trying all the possible keys.

$P = Dec_k(C)$ I don't know if K is correct, but I know the structure. I can get plaintext and look if first byte has structure (1st bit = parity, then 7 bits).
The probability of 1st bit being a parity bit is 0.5. This for all the bytes in the block.

$P\{\text{all bits represent a parity bit}\} = \left(\frac{1}{2}\right)^8$. This is the probability that a given key is a candidate key.

[expected] → $2^{56} \cdot \frac{1}{2^8} = 2^{48}$

Since $|K| = 2^{56}$, the number of candidate keys is $2^{56} \cdot \frac{1}{2^8} = 2^{48}$

So I need multiple blocks with the same key.
A candidate key with r blocks is a key that, take r blocks, will give us plaintexts with valid parity bits.

$P_r\{\text{this happens}\} = \frac{1}{2^8} \cdot \frac{1}{2^8} \cdots \frac{1}{2^8} = \frac{1}{2^{r \cdot 8}}$

So the number of candidate keys is $2^{56} \cdot \frac{1}{2^{8 \cdot r}} = 2^{56-8r}$ $r = 8, 9, 10$ is good.

[expected number of keys to make this decryption]

Exercise 2 - dictionary attack

- Consider a block cipher k and n.
- The adversary has collected D pairs (pt_i, ct_i), i = 1,…, D, with $D << 2^n$
- Now the adversary reads C newly produced cyphertexts ct^*_j, j = 1,…, C.
- Determine the value of C s.t. the Pr[Exists j, j = 1, 2,… C, s.t. c^*_j is in the dictionary] = P

Mar-25 Block Ciphers 20

20

Exercise 3 - Rekeying

- An adversary can successfully perform an exhaustive key search in a month.
- Our security policy requires that keys are changed every hour.
- What is the probability P that, in a month, the adversary is able to find any key before it is changed?
  - For simplicity assume that every month is composed of 30 days.
- What if we refresh key every minute?

Mar-25 Block Ciphers 21

21

Dictionary attack

Suppose we have a cypher with E, K, m.

Suppose attacker has a dictionary of D entries of ct and pt.

ct	pt
–	–
–	–
⋮	⋮
–	–

D

$D \ll 2^m$ ASSUMPTION

$A \xrightarrow{ct_i^*} B$

$\#\, ct_i^* = C$

A and B exchange ct_i (not necessarily is in the dictionary)

Determine the value of C, s.t. $P_r[\text{at least one } ct_i^* \in \text{Dictionary}] = ?$

[if they generate 100 messages, prob. that at least one of them is in the dictionary]

Let Q be the complementary event: $Q = 1 - P \Rightarrow P_r[Q] = P_r[\text{no cypherText is in the dic}]$

Let $q = P_r[ct_i^* \text{ is not in the Dictionary}] = \frac{2^m - D}{2^m} = 1 - \alpha$, $\alpha \triangleq \frac{D}{2^m} \ll 1$.

So for Q we apply pr. of the joint event: $Q = q^C = (1-\alpha)^C \simeq 1 - C\alpha$ *

*We neglect all the powers of α greater or eq. than 2, since α is very small

So, $P = 1 - Q \simeq C \cdot \alpha = C \cdot \frac{D}{2^m}$

If $P = 1$ → Be almost certain that we have one word in dictionary

$C = \frac{2^m}{D}$, $D = 2^{m/2}$, $C = 2^{m/2}$

PANKO PONDERS:

PANKO REASONS:

EX: $m = 64$ $\quad D = 2^{m/2} = 2^{32}$ → 8 bytes + 8 bytes

Size of dictionary $= 2^{32}(2^3 + 2^3) = 2^{32} \cdot 2^4 = 16$ Gbytes ← this is the traffic to change keys

Very likely that after 16 Gbytes of traffic encryption can be broken.

Rekeying

- We assume adversary can perform a bruteforce attack in 1 month (30 days)
- We have a security policy which tells us to rekey every hour

PANKO ASSUMES:

Compute $Pr[\text{Adversary is able to find at least 1 key in a month}] = P$

PANKO INQUIRES:

$H = 30 \cdot 24 = 720$ hours in a month (and so I generate 720 keys)

- ASSUME THAT CAPABILITY OF BRUTEFORCE IS UNIFORMLY DISTRIBUTED OVER THE MONTH

$$p = Pr[\text{a given key is guessed in 1 hour}] = \frac{1}{720} = 1.4 \cdot 10^{-3}$$

$$q = 1 - p = Pr[\text{a given key is not guessed in 1 hour}] = \frac{719}{720} = 1 - \frac{1}{H}$$

Again, let $\bar{P} = 1 - P = Pr[\text{Adversary cannot find any key}] = q^H = (1-p)^H = \left(1 - \frac{1}{H}\right)^H = \left(1 - \frac{1}{720}\right)^{720} =$

$= 0.37 \qquad P = 0.63$

Let us suppose now that we rekey every single minute: $H = 30 \cdot 24 \cdot 60$

We can extend the same reasoning as before to get more or less the same values.

This is because $\lim_{H \to \infty} \left(1 - \frac{1}{H}\right)^H = \frac{1}{e}$

- implication: beyond a certain limit it is not wise to increase rekeying rate
- So rekeying is useful (for what said before), but there is a limit

PANKO DEDUCES:

Symmetric Encryption

MULTIPLE ENCRYPTION AND KEY WHITENING

Increasing the Security of Block Ciphers

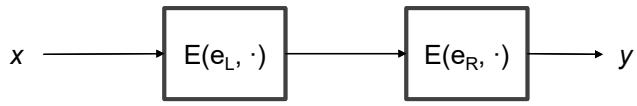
- DES is a secure cipher, no efficient cryptanalys is known Some vulnerabilities were discovered but were not exploitable in practice
- DES does not define a group
- DES key has become too short
- Can we improve the security of DES?
- Yes, by means of two techniques We use DES to encrypt plaintext multiple times. $P = E_{K2}(E_{K1}(P))$
  - Multiple encryption
  - Key whitening

DES does not define a group

- If DES were a group then $\forall k_1, k_2 \in \mathcal{K}, \exists k_3 \in \mathcal{K}$ s.t. $\forall x \in \mathcal{M}, E_{k_2}\left(E_{k_1}(x)\right) = E_{k_3}(x)$
- So, double (multiple) encryption would be useless
- Furthermore, DES would be vulnerable to Meet-in-the-Middle attack that runs in 2^{28}

Two-times Encryption (2E)

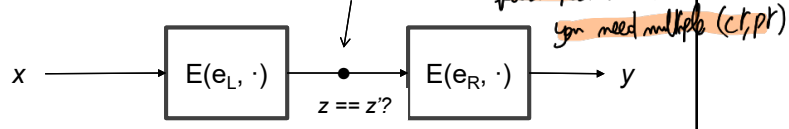
- $y = 2E((e_L, e_R), m) = E(e_R, E(e_L, x))$
  - key size is 2k bits
  - Brute force attack requires 2^{2k} steps
  - 2E is two times slower than E
- Is it really secure?
- Meet-in-the-middle attack



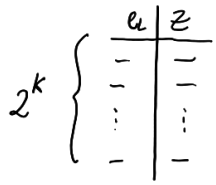
Meet-in-the-middle attack

- Attack Sketch
  1. Build a table T containing $z = E(e_L, x)$ for all possible keys e_L. Keep T sorted according to z.
  2. Check whether $z' = D(e_R, y)$ is contained in the table T, for all possible key e_R.
     1. If z' in contained in T then (e_L, e_R) maps x into y with e_L s.t. $T[e_L] = z'$.

Those are candidate keys. You may have false positives so you need multiple (c,p,r)



26

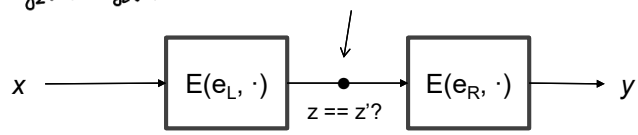


Meet-in-the-middle attack

- Attack complexity: data complexity: negligible
  - Storage complexity
    - Storage necessary for table $T \approx O(2^k)$
  - Time complexity
    - Time complexity for step 1 + Time complexity for step 2 = Time for building and sorting the table + Time for searching in a sorted table = $k\,2^k + k\,2^k \approx O(k2^k)$ NOT good enough

$\log_2(m) = \log_2(2^k) = k$

Meet-in-the-middle



27

Two-times DES

- 2DES
  - Time complexity: 2^{56} (doable nowadays!)
  - Space complexity: 2^{56} (lot of space!)
  - 2DES brings no advantage

Mar-25 Block Ciphers 28

28

Triple DES (3DES)

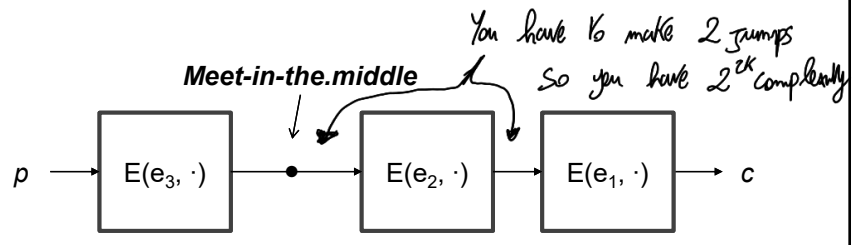
- EDE scheme
  - Standard ANSI X9.17 and ISO 8732
  - $Y = 3E((e_1, e_2, e_3), x) = E(e_1, D(e_2, E(e_3, x)))$
    - If $e_1 = e_2 = e_3$, 3DES becomes DES
      - backward compatibility
  - Key size = 168-bits
  - 3 times slower than DES
  - Simple attack ≈ 2^{118} more complexity to manage order in the table

Mar-25 Block Ciphers 29

29

3DES – meet-in-the-middle attack

- Time = 2^{112} (undoable!)
- Space = 2^{56} (lot of space!)



Mar-25 Block Ciphers 30

30

False positives for multiple encryption

- THEOREM
  - Given there are r subsequent encyptions with a block cipher with a key lenght of k bits and a block size of n bits, as well as t plaintext-ciphertext pairs, $(pt_1, ct_2),\ldots, (pt_t, ct_t)$, the expected number of false keys which encrypt all plaintext to the corresponsig ciphertext is 2^{rk-tn}

↳ like the keys are r times as long

Mar-25 Block Ciphers 31

31

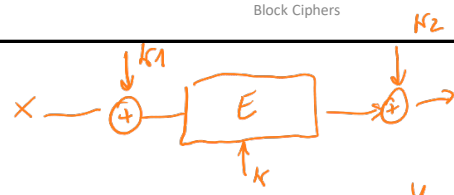
Limitations of 3DES

- 3DES resists brute force but
  - It is not efficient regarding software implementation
  - It has a short block size (n = 64)
    - A drawback if you want to make a hash function from 3DES, for example
  - Key lengths of ~~2000~~ are necessary to resist quantum computing attack

70s technology! Uses ops that work well in HW, not so well in SW

Not only for dictionary attacks. Ciphers are also used to implement hash functions, and 64 bits blocks make a hash not secure enough

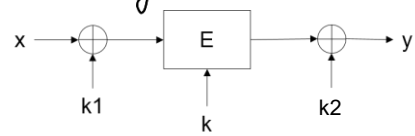
at least 256 bits



Key whitening

- Considerations
  - KW is not a "cure" for weak ciphers
- Applications
  - DESX: a variant of DES
  - AES: uses KW internally
- Performance
  - Negligible overhead w.r.t. E (Just two XOR's!)

Sort of 1TP



Definition 5.3.1 Key whitening for block ciphers
Encryption: $y = e_{k,k_1,k_2}(x) = e_k(x \oplus k_1) \oplus k_2$.
Decryption: $x = e^{-1}_{k,k_1,k_2}(x) = e^{-1}_k(y \oplus k_2) \oplus k_1$

Key whitening

- Attacks
  - Brute-force attack
    - Time complexity: 2^{k+2n} encryption ops
  - Meet-in-the-middle:
    - Time complexity 2^{k+n}
    - Storage complexity: 2^n data sets
  - The most efficient attack
    - If the adversary can collect 2^m pt-ct pairs, then time complexity becomes 2^{k+n-m}
      - The adversary cannot control m (rekeying)
    - Example: DES (m = 32)
      - Time complexity 2^{88} encryptions (nowadays, out of reach)
      - Storage complexity 2^{32} pairs = 64 GBytes of data (!!!)

Symmetric Encryption

ENCRYPTION MODES

Encryption Modes

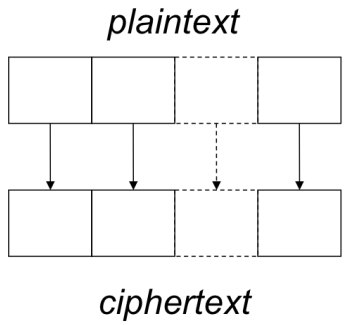
- A block cipher encrypts PT in fixed-size *n*-bit blocks
- When the PT len exceeds n bits, there are several modes to use the block cipher
  - Electronic Codebook (ECB) Recently deprecated by NIST
  - Cipher-block Chaining (CBC)

Other encryption modes

- Other encryption modes
  - To build a stream cipher out of a block cipher
    - Cipher Feedback mode (CFB)
    - Output Feedback mode (OFB)
    - Counter mode (CTR)
  - Authenticated encryption (encryption for confidentiality and integrity)
    - Galois Counter mode (GCM, CCM, …)
  - and many others (e.g., CTS, …)
- Block ciphers are very versatile components

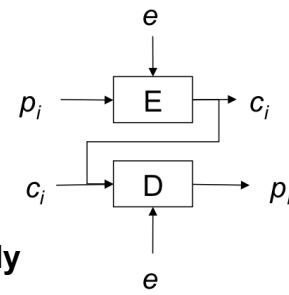
Electronic codebook

Assumption: size of the plaintext is a multiple of the size of the block



$$\forall 1 \le i \le t, c_i \leftarrow E(e, p_i)$$

$$\forall 1 \le i \le t, p_i \leftarrow D(e, c_i)$$



PT blocks are encrypted separately

38

ECB - properties

- PROS
  - No error propagation
    - One or more bits in a single CT block affects decryption of that block only
  - Enc & Dec can be parallelized
- CONS (it is insecure)
  - Blocks are encrypted separately
    - Identical PT results in identical CT
      - ECB doesn't hide data pattern
      - ECB allows traffic analysis
    - ECB allows block re-ordering and substitution

↑ a corruption affects only that PT

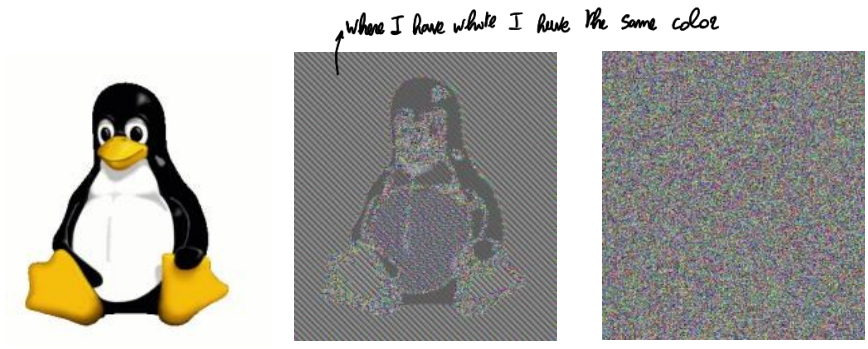
↳ given the same key

I can change orders of blocks and it doesn't affect decryption

Same for substitution

39

ECB doesn't hide data patterns



Plaintext | ECB encrypted | Non-ECB encrypted

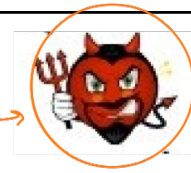
40

ECB – block attack

- Bank transaction that transfers a customer C's amount of money D from bank B1 to bank B2
  - Bank B1 debits D to C
  - Bank B1 sends the "credit D to C" message to bank B2
  - Upon receiving the message, Bank B2 credits D to C
- Credit message format
  - Src bank: M (12 byte)
  - Rcv banck: R (12 byte)
  - Customer: C (48 byte)
  - Bank account number: N (16 byte)
  - Amount of money: D (8 byte)
- Cipher: n = 64 bit; ECB mode

41

ECB – block attack



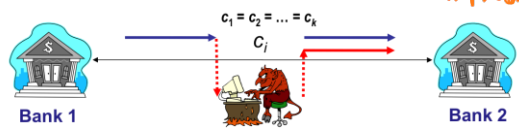
- Mr. Lou Cipher is a client of the banks and wants to make a fraud
- Attack aim
  - To replay Bank B1's message "credit 100$ to Lou Cipher" many times
- Attack strategy
  - Lou Cipher activates multiple transfers of 100$ so that multiple messages "credit 100$ to Lou Cipher" are sent from B1 to B2
  - The adversary identifies at least one of these messages
  - The adversary replies the message several times

Same message, if we use same keys, we have same cyphertext.

ECB – block attack

- The fraud
  1. Mr. Lou Cipher performs k equal transfers
     - credit 100$ to Lou Cipher → c1
     - credit 100$ to Lou Cipher → c2
     - ...
     - credit 100$ to Lou Cipher → c_k
  2. Then, he searches for "his own" CTs, namely k equal CTs!
  3. Finally he replies one of these cryptograms (many times)

↑ to recognise with higher probability our right frame



↳ problem as that you messages you have no notion of time, so ...

ECB – block attack

- The message lacks any notion of time so it can be easily replied
- An 8-byte timestamp field T (block #1) is added to the message to prevent replay attacks
- A replied message can now be discarded

Mar-25 Block Ciphers 44

44

ECB – block attack

- However, Mr Lou Cipher can still perform the attack
  1. Identify "his own" CTs by inspecting blocks #2-#13
  2. Select any his-own-CT
  3. Substitute block #1 of his-own-CT with block #1 of any intercepted "fresh" block
  4. Replay the resulting CT

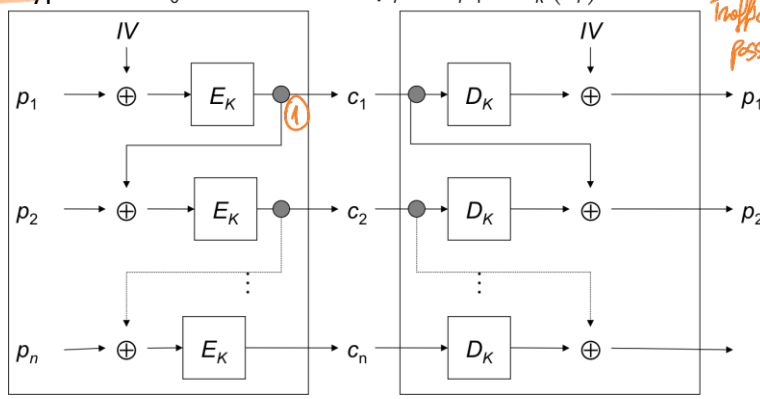
Mar-25 Block Ciphers 45

45

Cipher block chaining (CBC)

Encryption: $c_0 \leftarrow IV. \forall 1 \le i \le t, c_i \leftarrow E_k(p_i \oplus c_{i-1})$

Decryption: $c_0 \leftarrow IV. \forall 1 \le i \le t, p_i \leftarrow c_{i-1} \oplus D_k(c_i)$



① can be seen as random from a limited adversary (like TTP)
If I transmit same message but use different IV, done. Traffic analysis not possible. And I cannot insert blocks.

still under the assumption that message is an n-key multiple of the block size

46

CBC – properties (→)

- CBC mode is CPA-secure
- CBC-Enc is *randomized* by using IV (nonce) [IV has to change]
  - Identical ciphertext results from the same PT under the same key and IV (IV diff., traffic analysis not possible)
- Chaining dependencies: c_i depends on p_i and the preceding PT block
- CT-block reordering affects decryption (decryption fails) → same for substitution
- Cyphertext expansion is just one block

47