

"The best system is to use a simple, well understood algorithm which relies on the security of a key rather than the algorithm itself. This means if anybody steals a key, you could just roll another and they have to start all over."
-- Andrew Carol

Symmetric key cryptography

- preliminaries (operational and attacker models)
- block ciphers (basics, DES, 3DES, AES)
- block ciphers in practice (modes of operation)
- a security flaw induced by CBC padding
- stream ciphers

Large numbers

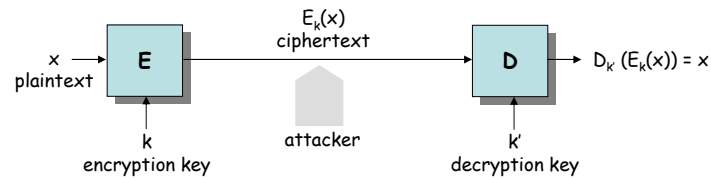
time until next ice age..... 2^{39} seconds
time until the sun goes nova..... 2^{55} seconds
age of the planet..... 2^{55} seconds
age of the Universe..... 2^{59} seconds

number of atoms in the planet..... 2^{170}
number of atoms in the sun..... 2^{190}
number of atoms in the galaxy..... 2^{223}
number of atoms in the Universe 2^{265}
(dark matter excluded)

volume of the universe..... 2^{280} cm³

(source: Schneier, Applied Cryptography, 2nd ed., Wiley 1996)

Operational model of encryption



- attacker's goal:
 - to systematically recover plaintext from ciphertext
 - to deduce the (decryption) key
- Kerckhoff's assumption:
 - attacker knows all details of E and D
 - attacker doesn't know the (decryption) key

© Levente Buttyán

3

Attack models

- ciphertext-only
 - only data transmitted over the ciphertext channel is available to the attacker
- known-plaintext
 - plaintext-ciphertext pairs are available to the attacker
- chosen-plaintext
 - ciphertexts are available corresponding to plaintexts of the attacker's choice
 - adaptive: choice of plaintexts may depend on previously obtained plaintext-ciphertext pairs
- chosen-ciphertext
 - plaintext-ciphertext pairs are available for some number of ciphertexts of the attacker's choice
 - adaptive: choice of ciphertexts may depend on previously obtained plaintext-ciphertext pairs
- related-key attack
 - attacker has access to the encryption of plaintexts under both the unknown key and keys known to have certain relationship with the unknown key

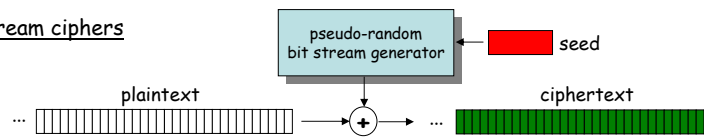
© Levente Buttyán

4

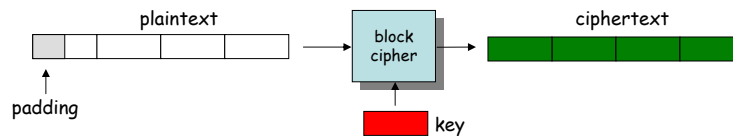
Asymmetric- vs. symmetric-key encryption

- asymmetric-key encryption
 - it is hard (computationally infeasible) to compute k' from k
 - k can be made public (public-key cryptography)
- symmetric-key encryption
 - it is easy to compute k from k' (and vice versa)
 - often $k = k'$
 - two main types: stream ciphers and block ciphers

stream ciphers



block ciphers



© Levente Buttyán

5

Theoretical vs. practical security

- one-time pad
 - a stream cipher where the key stream is a true random bit stream
 - unconditionally secure [Shannon, 1949]
 - however, the key must be as long as the plaintext to be encrypted
- practical ciphers
 - use much shorter keys (128 bits (symm.), 1024 bits (asymm.))
 - they are not unconditionally secure, but computationally infeasible to break (practically secure)
 - however, proving that a cipher is practically secure is not easy
 - not enough to consider *brute force attacks* (key size) only
 - a cipher may be broken due to weaknesses in its (algebraic) structure
 - no proofs of security exist for many ciphers used in practice
 - if a proof exists, it usually relies on assumptions that are widely believed to be true (such as $P \neq NP$)

© Levente Buttyán

6

Attack complexity

- best available measure of security for practical ciphers is the complexity of the best (currently) known attack
- attack complexity is the dominant of the following complexities
 - data complexity
 - expected number of input data units required for the attack (e.g., number of ciphertexts in a ciphertext-only attack)
 - storage complexity
 - expected number of storage units required
 - processing complexity
 - expected number of operations required to process input data and/or fill storage with data
 - parallelization may reduce attack time but not processing complexity

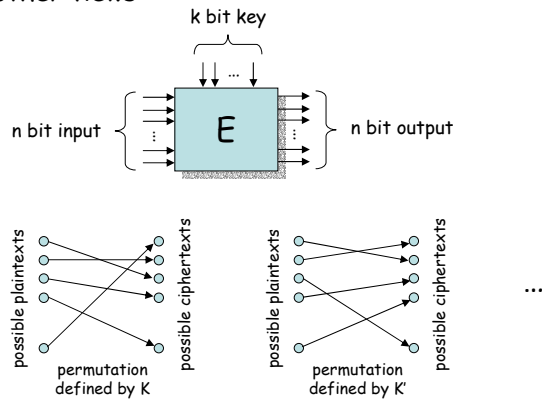
"Feistel and Coppersmith rule: Sixteen rounds and one hell of an avalanche."
-- Stephan Eisvogel in de.comp.security

Block ciphers: DES, 3DES, and AES

- basics
- operation of DES
- cryptanalysis of DES
- multiple encryption and the 3DES
- AES

Block ciphers

- an n bit block cipher is a function $E: \{0, 1\}^n \times \{0, 1\}^k \rightarrow \{0, 1\}^n$, such that for each $K \in \{0, 1\}^k$, $E(X, K) = E_K(X)$ is an invertible mapping from $\{0, 1\}^n$ to $\{0, 1\}^n$
- the inverse of $E_K(X)$ is denoted by $D_K(Y)$, where $Y = E_K(X)$
- two other views:



© Levente Buttyán

9

Trade-offs in block size

- attacks based on small block size
 - text dictionary attack
 - plaintext-ciphertext pairs become known for a fixed key
 - the "larger" the dictionary the greater the chance of locating a random ciphertext in it
 - if n is small, then it is feasible to build "large" dictionaries
 - matching ciphertext attacks
 - if a dictionary of size about $2^{n/2}$ have been created, and about $2^{n/2}$ ciphertexts are subsequently given, then one expects to locate a ciphertext in the dictionary with high probability (birthday paradox)
- larger block size is more secure
- disadvantages of large block size
 - more costly to implement (in terms of gates or low level instructions)

© Levente Buttyán

10

Exhaustive key search and key size

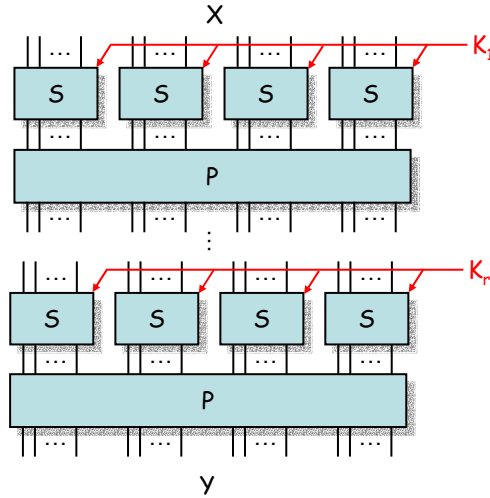
- given a small number of plaintext-ciphertext pairs encrypted under a key K , K can be recovered by exhaustive key search with 2^{k-1} processing complexity (expected number of operations)
 - input: $(X, Y), (X', Y'), \dots$
 - progress through the entire key space
 - for each trial key K' , decrypt Y
 - if the result is not X , then throw away K'
 - if the result is X , then check the other pairs $(X', Y'), \dots$
 - if K' does not work for at least one pair, then throw away K'
 - if K' worked for all pairs $(X, Y), (X', Y'), \dots$, then output K' as the target key
 - on average, the target key is found after searching half of the key space
- if the plaintexts are known to contain redundancy, then ciphertext-only exhaustive key search is possible with a relatively small number of ciphertexts

How to build a “strong” block cipher?

- complex encryption function can be built by composing several simple operations which offer complementary - but individually insufficient - protection
- simple operations:
 - elementary arithmetic operations
 - logical operations (e.g., XOR)
 - modular multiplication
 - transpositions
 - substitutions
 - etc.
- let's combine two or more transformations in a manner that the resulting cipher is more secure than the individual components

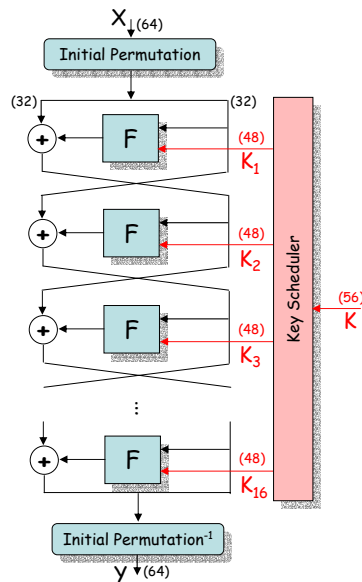
Example: SP networks

- an SP (substitution-permutation) network is a *product cipher* composed of stages each involving key controlled substitutions (non-linear look-up tables) and permutations



© Levente Buttyán

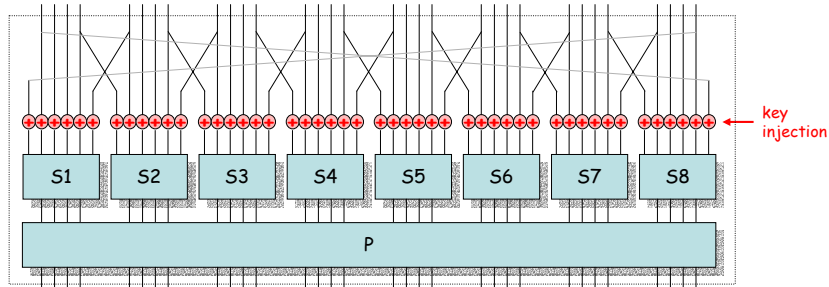
DES - Data Encryption Standard



- input size: 64
- output size: 64
- key size: 56
- 16 rounds
- Feistel structure

© Levente Buttyán

DES round function F

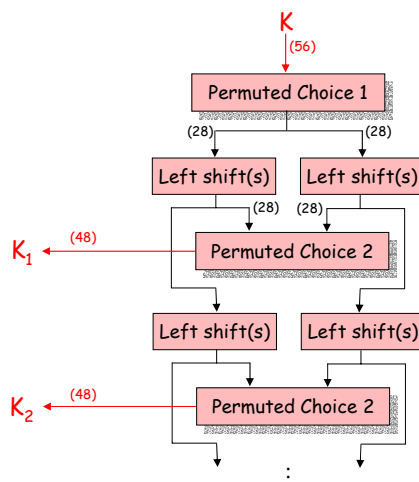


- S_i - substitution box (S-box)
- P - permutation box (P-box)

© Levente Buttyán

15

DES key scheduler

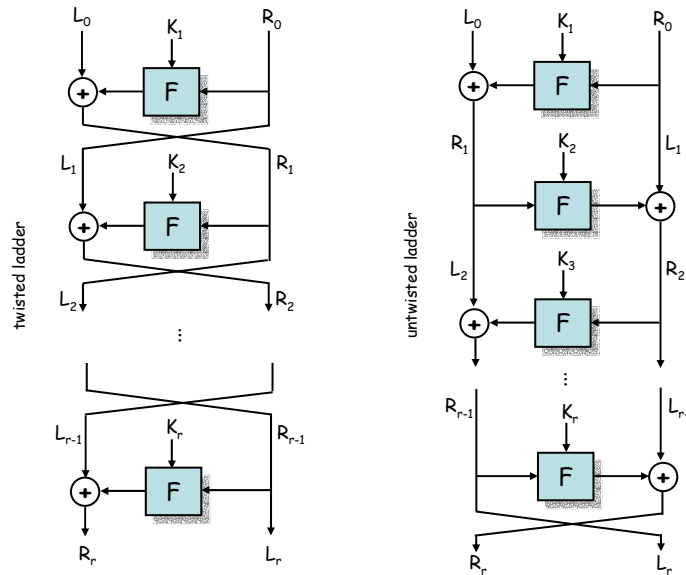


- each key bit is used in around 14 out of 16 rounds

© Levente Buttyán

16

Feistel structure illustrated



© Levente Buttyán

17

Properties of Feistel ciphers

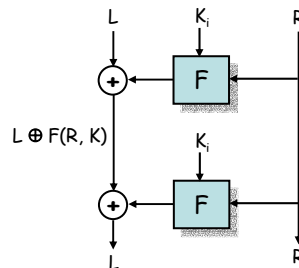
- round i maps (L_{i-1}, R_{i-1}) into (L_i, R_i) as follows:

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$
- a Feistel cipher is always invertible even if F is not invertible:

$$R_{i-1} = L_i$$

$$L_{i-1} = R_i \oplus F(R_{i-1}, K_i) = R_i \oplus F(L_i, K_i)$$
- decryption can be achieved using the same r -round process with the round keys used in reverse order (K_r through K_1)

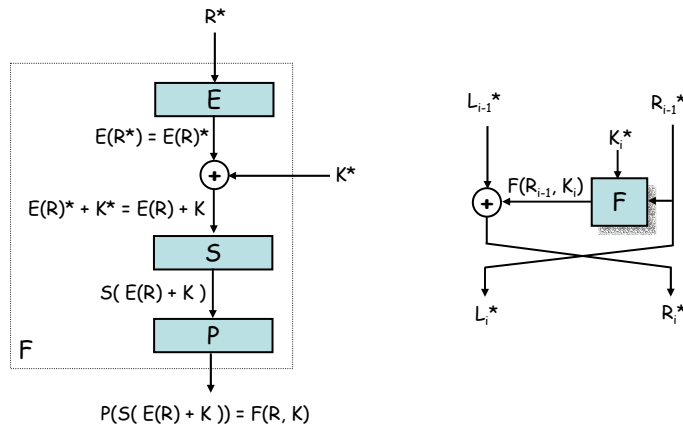


© Levente Buttyán

18

Complementation property of DES

- $Y = \text{DES}_K(X)$ implies $Y^* = \text{DES}_{K^*}(X^*)$
 - where X^* denotes the bitwise complement of X



© Levente Buttyán

19

Consequences of the complementation prop.

- assume an attacker can mount a chosen-plaintext attack
- the attacker chooses a plaintext X , and obtains $Y_1 = \text{DES}_K(X)$ and $Y_2 = \text{DES}_K(X^*)$
- by the complementation property, the attacker knows that $\text{DES}_{K^*}(X) = Y_2^*$
- the attacker then runs an exhaustive key search
 - for each trial key K' , he computes $Y' = \text{DES}_{K'}(X)$
 - if $Y' = Y_1$, then K' is possibly the target key (should be further tested)
 - if $Y' = Y_2^*$, then K'^* is possibly the target key (should be further tested)
 - otherwise throw away both K' and K'^*
- expected number of keys required before success is reduced from 2^{55} to 2^{54}
- still impractical as an attack

© Levente Buttyán

20

DES weak keys and semi-weak keys

- a weak key is a key K such that $DES_K(DES_K(X)) = X$
 - there are 4 DES weak keys:
0101 0101 0101 0101
FEFE FEFE FEFE FEFE
1F1F 1F1F 0E0E 0E0E
E0E0 E0E0 F1F1 F1F1
- a semi-weak key pair is a pair (K_1, K_2) such that $DES_{K_1}(DES_{K_2}(X)) = X$
 - there are 6 pairs of DES semi-weak keys
- why are these keys weak?
 - for each weak key K , there exist 2^{32} fix points of DES_K , i.e., plaintext X such that $DES_K(X) = X$
 - for 4 out of the 12 semi-weak keys, there exist 2^{32} anti-fix points, i.e., plaintext X such that $DES_K(X) = X^*$

© Levente Buttyán

21

Linear and differential cryptanalysis of DES

- linear cryptanalysis (LC)
 - linear cryptanalysis is the most powerful attack against DES to date
 - requires an enormous number ($\sim 2^{43}$) known plaintext-ciphertext pairs \rightarrow infeasible in practical environments
 - could work in a ciphertext only model if plaintexts are redundant (e.g., contain parity bits)
- differential cryptanalysis (DC)
 - most general cryptanalytic tool to date against iterated block ciphers (including DES, FEAL, IDEA)
 - primarily a chosen-plaintext attack
 - in case of DES, it requires $\sim 2^{47}$ chosen plaintext-ciphertext pairs \rightarrow infeasible in practical environments
- DES was optimized against DC when it was designed
- it can, however, be improved with respect to LC (apparently the designers of DES was not aware of this attack at that time)

© Levente Buttyán

22

The basic idea of DC

- consider a single S-box S_i of DES
- fix an input difference Δx
- determine the distribution of the output differences

$$\text{DDT}_{S_i}(\Delta x, \Delta y) = |\{ x : S_i(x) \oplus S_i(x \oplus \Delta x) = \Delta y \}|$$
- this distribution will not be uniform !!!
- example:

S_1 and $\Delta x = 110100$

Δy	$\text{DDT}_{S_1}(\Delta x, \Delta y)$	Δy	$\text{DDT}_{S_1}(\Delta x, \Delta y)$
0000	0	1000	6
0001	8	1001	0
0010	16	1010	0
0011	6	1011	0
0100	2	1100	0
0101	0	1101	8
0110	0	1110	0
0111	12	1111	6

© Levente Buttyán

23

The basic idea of DC

- let $\text{IN}_{S_i}(\Delta x, \Delta y) = \{ x : S_i(x) \oplus S_i(x \oplus \Delta x) = \Delta y \}$
 - note: $\text{DDT}_{S_i}(\Delta x, \Delta y) = |\text{IN}_{S_i}(\Delta x, \Delta y)|$
- example: S_1 and $\Delta x = 110100$

Δy	$\text{IN}_{S_1}(\Delta x, \Delta y)$	$\text{DDT}_{S_1}(\Delta x, \Delta y)$
0000		0
0001	000011, 001111, 011110, 011111, 101010, 101011, 110111, 111011	8
0010	000100, 000101, 001110, 010001, 010010, 010100, 011010, 011011, 100000, 100101, 010110, 101110, 101111, 110000, 110001, 111010	16
0011	000001, 000010, 010101, 100001, 110101, 110110	6
0100	010011, 100111	2
0101		0
0110		0
0111	000000, 001000, 001101, 010111, 011000, 011101, 100011, 101001, 101100, 110100, 111001, 111100	12
1000	001001, 001100, 011001, 101101, 111000, 111101	6
1001		0
1010		0
1011		0
1100		0
1101	000110, 010000, 010110, 011100, 100010, 100100, 101000, 110010	8
1110		0
1111	000111, 001010, 001011, 110011, 111110, 111111	6

© Levente Buttyán

24

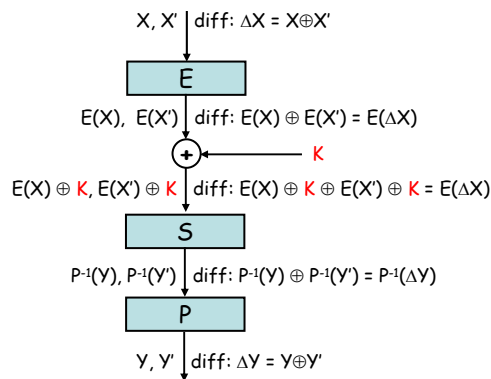
The basic idea of DC

- there are 64 possible input differences (Δx) and 8 S-boxes
 \rightarrow there are $64 \cdot 8 = 512$ such tables
- these can be computed easily with a computer
- assume that x and x' are input to S_i and the results are y and y'
- assume that we don't know x , x' , y , and y' , but we know $\Delta x = x \oplus x'$ and $\Delta y = y \oplus y'$
- then the possible values of x (and x') are listed in $IN_{S_i}(\Delta x, \Delta y)$



© Levente Buttyán

Breaking the round function F



- assume we know $X, X', \Delta Y$
- then we know the input and output differences of each S-box
 \rightarrow if $K = K_1 K_2 \dots K_8$, then $E(X)_i \oplus K_i \in IN_{S_i}(E(\Delta X)_i, P^{-1}(\Delta Y)_i)$
 $\rightarrow K_i \in \{ B \oplus E(X)_i : B \in IN_{S_i}(E(\Delta X)_i, P^{-1}(\Delta Y)_i) \}$ for all $i = 1, 2, \dots, 8$

© Levente Buttyán

Breaking the round function F

- let $T_i(X, X', \Delta Y) = \{ B \oplus E(X)_i : B \in \text{IN}_{S_i}(E(X \oplus X'), P^{-1}(\Delta Y)_i) \}$
- given a triplet $X, X', \Delta Y$, we know that $K_i \in T_i(X, X', \Delta Y)$ for every i
- if we have J triplets, then we know that for every i

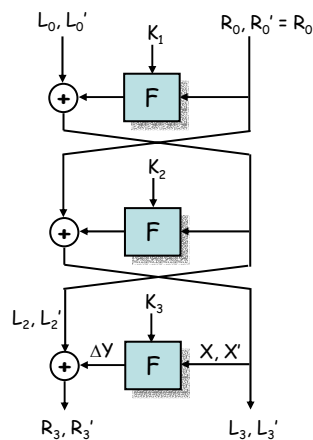
$$K_i \in \bigcap_{j=1}^J T_i(X^{(j)}, X'^{(j)}, \Delta Y^{(j)})$$

- only the right value of K_i will appear in every $T_i(X^{(j)}, X'^{(j)}, \Delta Y^{(j)})$!!!
- keep 64 counters for every K_i ($i = 1, 2, \dots, 8 \rightarrow 512$ counters)
- increment counter v if $v \in T_i(X^{(j)}, X'^{(j)}, \Delta Y^{(j)})$ for some j
- at the end, there will be a single counter v^* with value J for every $i \rightarrow K_i = v^*$

© Levente Buttyán

27

Breaking the 3-round DES

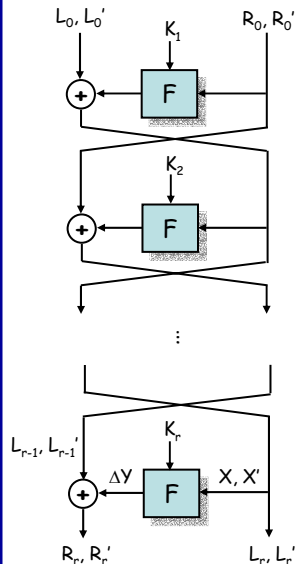


- consider the round function of the last round:
 $X = L_3$
 $X' = L_3'$
 $\Delta Y = (R_3 \oplus L_2) \oplus (R_3' \oplus L_2')$
 $= R_3 \oplus R_3' \oplus L_0 \oplus F(R_0, K_1) \oplus L_0' \oplus F(R_0, K_1)$
 $= R_3 \oplus R_3' \oplus L_0 \oplus L_0'$
- if we know several $(L_0 R_0, R_3 L_3)$ pairs (such that R_0 is the same for all pairs), then we can determine K_3
- the remaining 8 bits of the key can be determined with exhaustive key search

© Levente Buttyán

28

What about more than 3 rounds ?



© Levente Buttyán

- consider the round function of the last round:

$$\begin{aligned} X &= L_3 \\ X' &= L_3' \\ \Delta Y &= (R_r \oplus L_{r-1}) \oplus (R_r' \oplus L_{r-1}') \\ &= R_r \oplus R_r' \oplus L_{r-1} \oplus L_{r-1}' \\ &= \Delta R_r \oplus \Delta L_{r-1} = ? \end{aligned}$$

- it is possible to find difference pairs $(\Delta P, \Delta C)$ such that $\Pr\{\Delta L_{r-1} R_{r-1} = \Delta C \mid \Delta L_0 R_0 = \Delta P\} > 2^{-n}$
 - algorithm:
 - choose many input pairs with difference ΔP , and ask for their encryption (chosen plaintext attack)
 - for each pair, assume that the difference before the last round is ΔC , and
 - determine the set of possible values for K_r
 - if your assumption was correct, then the right value will surely be in the set of possible values
 - if your assumption was incorrect, then every key appears in the set with uniform probability
- the right value of K_r will appear most frequently

29

Preliminaries for LC

- Piling-up lemma:

Let X_1, X_2, \dots, X_n be independent random variables, such that $X_i \in \{0, 1\}$ and $\Pr\{X_i = 0\} = \frac{1}{2} + q_i$. Then

$$\Pr\{X_1 \oplus X_2 \oplus \dots \oplus X_n = 0\} = \frac{1}{2} + 2^{n-1} q_1 q_2 \dots q_n$$

Proof: BOOK, page 60.

- notations:

\oplus - bitwise XOR

\otimes - binary scalar product: $x \otimes y = x_1 y_1 \oplus x_2 y_2 \oplus \dots \oplus x_n y_n$

- Linear Approximation Table: for an m bit to n bit S-box S , let

$$\text{LAT}_S(a, b) = |\{x : a \otimes x = b \otimes S(x)\}| - 2^{m-1}$$

© Levente Buttyán

30

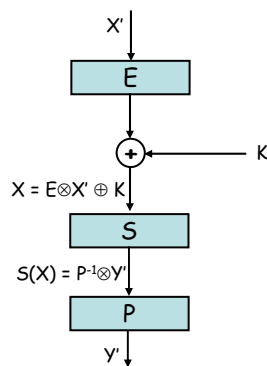
Linear approximations

- for an S-box S :
 - $a \otimes x \oplus b \otimes S(x) = 0$
 - with probability
 - $2^{-m} |\{ x : a \otimes x \oplus b \otimes S(x) = 0 \}| =$
 - $2^{-m} (\text{LAT}_S(a, b) + 2^{m-1}) =$
 - $\frac{1}{2} + 2^{-m} \text{LAT}_S(a, b) =$
 - $\frac{1}{2} + q$
- for an $\underline{S} = (S_1, S_2, \dots, S_k)$ S-box layer:
 - assuming that $a_i \otimes x \oplus b_i \otimes S_i(x) = 0$ with prob. $\frac{1}{2} + q_i$
 - we have
 - $A \otimes X \oplus B \otimes \underline{S}(X) = 0$ with prob. $\frac{1}{2} + 2^{k-1} q_1 q_2 \dots q_k = \frac{1}{2} + Q$
 - where
 - $A = (a_1 a_2 \dots a_k)$
 - $B = (b_1 b_2 \dots b_k)$

© Levente Buttyán

31

Linear approximations

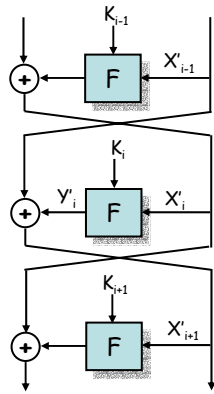


- for the round function F :
 - $X = E \otimes X' \oplus K$
 - $S(X) = P^{-1} \otimes Y'$
- using the approximation of the S-box layer:
 - $A \otimes (E \otimes X' \oplus K) \oplus B \otimes P^{-1} \otimes Y' = 0$
 - $(A \otimes E) \otimes X' \oplus (B \otimes P^{-1}) \otimes Y' \oplus A \otimes K = 0$
 - $A' \otimes X' \oplus B' \otimes Y' \oplus G \otimes K = 0$
 - with prob. $\frac{1}{2} + Q$

© Levente Buttyán

32

Linear approximations



- for the whole cipher:
 - let us assume that for every round i

$$A'_i \otimes X'_i \oplus B'_i \otimes Y'_i \oplus G_i \otimes K_i = 0$$
 with prob. $\frac{1}{2} + Q_i$
 - due to the Feistel structure we have
$$Y'_i = X'_{i-1} \oplus X'_{i+1} \quad (i = 2, \dots, r-1)$$

$$Y'_1 = P_L \oplus X'_2$$
 - summing up the first $r-1$ approximations
$$A'_1 \otimes X'_1 \oplus B'_1 \otimes (P_L \oplus X'_2) \oplus$$

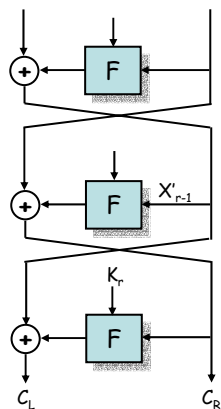
$$\bigoplus_{i=2, \dots, r-1} (A'_i \otimes X'_i \oplus B'_i \otimes (X'_{i-1} \oplus X'_{i+1})) \oplus$$

$$\bigoplus_{i=1, \dots, r-1} G_i \otimes K_i = 0$$
 with prob. $\frac{1}{2} + 2^{r-2} Q_1 Q_2 \dots Q_{r-1} = \frac{1}{2} + Q$

© Levente Buttyán

33

Linear approximations



- for the whole cipher (cont'd):

$$B'_1 \otimes P_L \oplus (A'_1 \oplus B'_2) \otimes X'_1 \oplus$$

$$\bigoplus_{i=2, \dots, r-2} (B'_{i-1} \oplus A'_i \oplus B'_{i+1}) \otimes X'_i \oplus$$

$$(B'_{r-2} \oplus A'_{r-1}) \otimes X'_{r-1} \oplus$$

$$\bigoplus_{i=1, \dots, r-1} G_i \otimes K_i = 0$$
 - assuming that $B'_{i-1} \oplus A'_i \oplus B'_{i+1} = 0$ for all $i = 2, \dots, r-2$, and using $X'_1 = P_R$ and $X'_{r-1} = C_L \oplus F(C_R, K_r)$, we get:
$$\pi \otimes P \oplus \gamma \otimes C \oplus \phi \otimes F(C_R, K_r) = \kappa \otimes K$$
 with prob. $\frac{1}{2} + Q$

© Levente Buttyán

34

Algorithm of LC

- consider a linear approximation

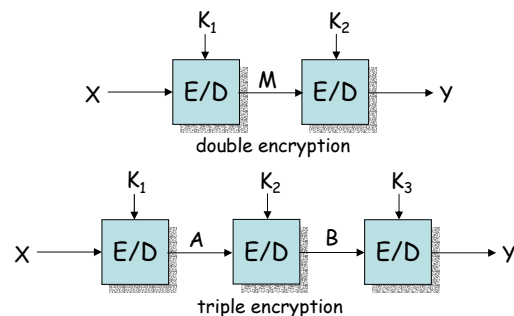
$$\pi \otimes P \oplus \gamma \otimes C \oplus \phi \otimes F(C_R, K_r) = \kappa \otimes K$$
 which holds with prob. $\frac{1}{2} + Q$
- take N plaintext-ciphertext pairs (P, C) (known plaintext attack)
- for every possible value v of K_r
 - compute $\pi \otimes P \oplus \gamma \otimes C \oplus \phi \otimes F(C_R, v)$ for every (P, C) pair
 - let T_v be the number of pairs for which the result is 0
 - if v is the wrong value, then the result is 0 for about half of the pairs $\rightarrow |T_v - N/2| \sim 0$
 - if v is the right value, then the result is $\kappa \otimes K$ for about $N(\frac{1}{2} + Q)$ pairs
 - if $\kappa \otimes K = 0$, then $T_v \sim N(\frac{1}{2} + Q)$
 - if $\kappa \otimes K = 1$, then $T_v \sim N(\frac{1}{2} - Q)$
 - in any case $|T_v - N/2| \sim |NQ|$
- select the value v^* for which $|T_{v^*} - N/2|$ is maximal
- output $K_r = v^*$

© Levente Buttyán

35

Multiple encryption and 3DES

- if a block cipher is susceptible to exhaustive key search (e.g., DES), then encryption of the same message more than once may increase security



- stage keys may not be independent
 - e.g., two-key 3DES: $K_1 = K_3$
- a stage cipher may be either a block cipher or its corresponding decryption function
 - e.g., 3DES-EDE (encryption-decryption-encryption)

© Levente Buttyán

36

DES is not a group

- group property
 - given any two keys K_1 and K_2 , there exists a third key K_3 , such that $DES_{K_1}(DES_{K_2}(X)) = DES_{K_3}(X)$ for all X
- if DES was a group, then multiple encryption would be equivalent to single encryption \rightarrow approaches like 3DES to strengthen DES would be useless

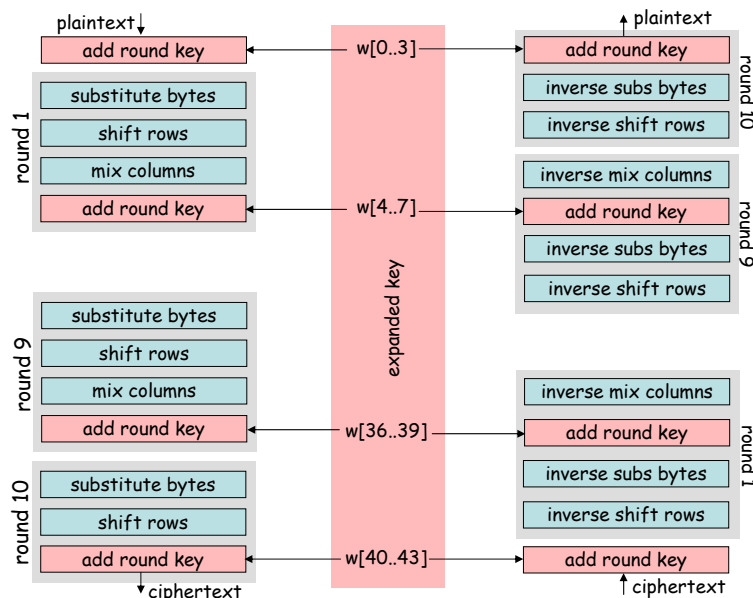
Meet-in-the-middle attack on double enc.

- a naïve exhaustive key search attack on double encryption tries all 2^{2k} keys
- a known-plaintext meet-in-the-middle attack defeats double encryption using an order of 2^k operations and 2^k storage
 - attack time is reduced at the cost of substantial space
- meet-in-the-middle attack:
 - input: known plaintext-ciphertext pairs $(X, Y), (X', Y'), \dots$
 - compute $M_i = E_i(X)$ for all possible key values $K_1 = i$ and store all (M_i, i) pairs in a table
 - compute $M'_j = D_j(Y)$ for all possible key values $K_2 = j$ and check for hits $M'_j = M_i$ against entries in the stored table
 - M'_j need not be stored, it can be checked as it is generated
 - each hit identifies a candidate solution key pair (i, j)
 - using a second plaintext-ciphertext pair (X', Y') , discard false hits
 - for an L stage cascade of random ciphers, the expected number of false key hits when t plaintext-ciphertext pairs are available is 2^{Lk-tn} , where n and k are the block and key sizes, resp.

AES - Advanced Encryption Standard

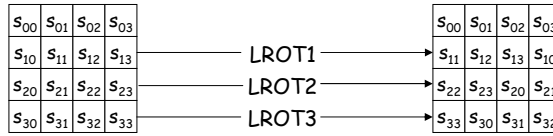
- NIST selected Rijndael (designed by Joan Daemen and Vincent Rijmen) as a successor of DES (3DES) in November 2001
- Rijndael parameters
 - key size 128 192 256
 - input/output size 128 128 128
 - number of rounds 10 12 14
 - round key size 128 128 128
- not Feistel structure
- decryption algorithm is different from encryption algorithm (optimized for encryption)
- single 8 bit to 8 bit S-box
- key injection (bitwise XOR)

General structure of encryption/decryption

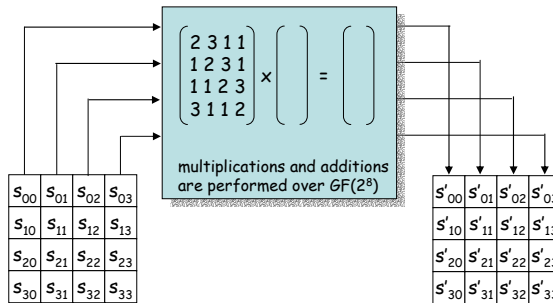


Shift row and mix column

shift row

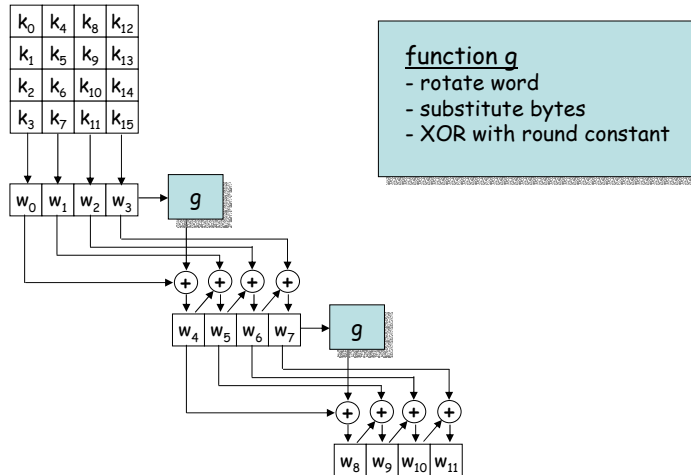


mix column



© Levente Buttyán

Key expansion



function g
 - rotate word
 - substitute bytes
 - XOR with round constant

© Levente Buttyán

Summary

- block cipher basics
 - trade-offs in block size
 - trade-offs in key size, exhaustive key search
 - product ciphers, SP networks
- DES
 - operation
 - properties (Feistel structure, complementation, weak keys)
 - differential and linear cryptanalysis
 - multiple encryption and the 3DES
 - meet-in-the middle attack on 2DES
- AES
 - operation

© Levente Buttyán

43

Using a block cipher in practice: modes of operation

- ECB
- CBC
- CFB, OFB, CTR

Block cipher modes of operation

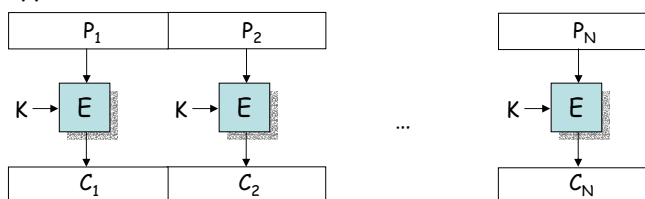
- ECB - Electronic Codebook
 - used to encipher a single plaintext block (e.g., a DES key)
- CBC - Cipher Block Chaining
 - repeated use of the encryption algorithm to encipher a message consisting of many blocks
- CFB - Cipher Feedback
 - used to encipher a stream of characters, dealing with each character as it comes
- OFB - Output Feedback
 - another method of stream encryption, used on noisy channels
- CTR - Counter
 - simplified OFB with certain advantages
- triple-inner-CBC and triple-outer-CBC
 - possible modes of operation when multiple encryption is used

© Levente Buttyán

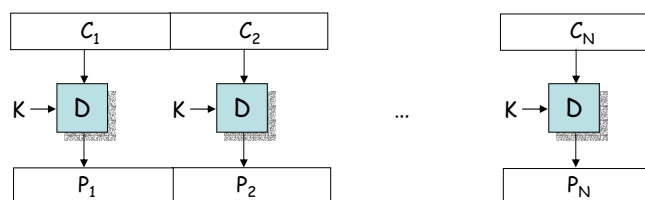
45

ECB mode

- encrypt



- decrypt



© Levente Buttyán

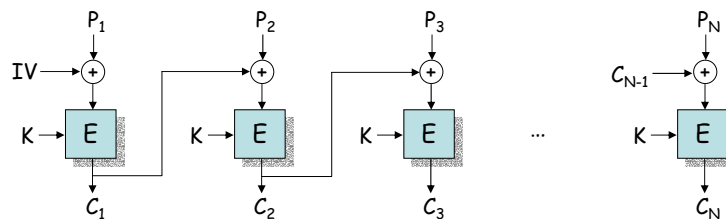
46

Properties of ECB mode

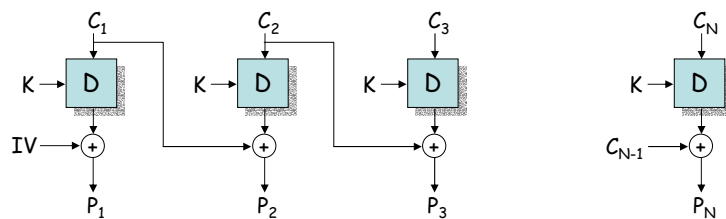
- identical plaintext blocks result in identical ciphertext blocks (under the same key of course)
 - messages to be encrypted often have very regular formats
 - repeating fragments, special headers, string of 0s, etc. are quite common
- blocks are encrypted independently of other blocks
 - reordering ciphertext blocks result in correspondingly reordered plaintext blocks
 - ciphertext blocks can be cut from one message and pasted in another, possibly without detection
- error propagation: one bit error in a ciphertext block affects only the corresponding plaintext block (results in garbage)
- overall: not recommended for messages longer than one block, or if keys are reused for more than one block

CBC mode

- encrypt



- decrypt



Properties of CBC mode

- encrypting the same plaintexts under the same key, but different IVs result in different ciphertexts
- ciphertext block C_j depends on P_j and all preceding plaintext blocks
 - rearranging ciphertext blocks affects decryption
 - however, dependency on the preceding plaintext blocks is only via the previous ciphertext block C_{j-1}
 - proper decryption of a correct ciphertext block needs a correct preceding ciphertext block only
- error propagation:
 - one bit error in a ciphertext block C_j has an effect on the j -th and $(j+1)$ -st plaintext block
 - P_j is complete garbage and P_{j+1} has bit errors where C_j had
 - an attacker may cause predictable bit changes in the $(j+1)$ -st plaintext block
- error recovery:
 - recovers from bit errors (self-synchronizing)
 - cannot, however, recover from frame errors ("lost" bits)

© Levente Buttyán

49

Integrity of the IV in CBC mode

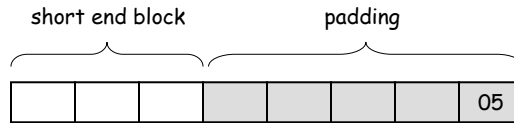
- the IV need not be secret, but its integrity should be protected
 - malicious modification of the IV allows an attacker to make predictable changes to the first plaintext block recovered
- one solution is to send the IV in an encrypted form at the beginning of the CBC encrypted message

© Levente Buttyán

50

Padding

- the length of the message may not be a multiple of the block size of the cipher
- one can add some extra bytes to the short end block until it reaches the correct size - this is called padding
- usually the last byte indicates the number of padding bytes added - this allows the receiver to remove the padding



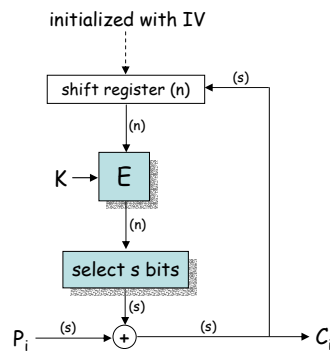
- note: if the encrypted message must have the same size as the clear message, then no padding can be used
 - encrypt the last ciphertext block again
 - select m bits and XOR them to the remaining m bits of the clear message

© Levente Buttyán

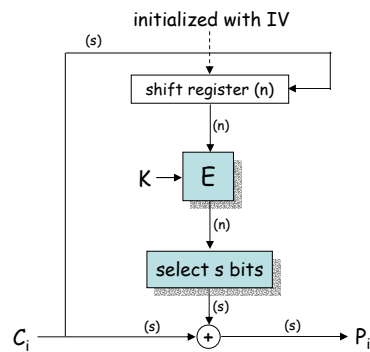
51

CFB mode

- encrypt



- decrypt



© Levente Buttyán

52

Properties of CFB mode

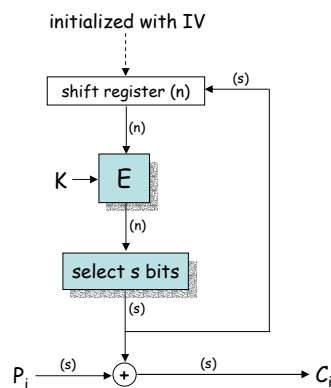
- encrypting the same plaintexts under the same key, but different IVs result in different ciphertexts
- the IV can be sent in clear
- ciphertext block C_j depends on P_j and all preceding plaintext blocks
 - rearranging ciphertext blocks affects decryption
 - proper decryption of a correct ciphertext block needs the preceding n/s ciphertext blocks to be correct
- error propagation:
 - one bit error in a ciphertext block C_j has an effect on the decryption of that and the next n/s ciphertext blocks (the error remains in the shift register for n/s steps)
 - P_j' has bit errors where C_j had, all the other erroneous plaintext blocks are garbage
 - an attacker may cause predictable bit changes in the j -th plaintext block
- error recovery:
 - self synchronizing, but requires n/s blocks to recover

© Levente Buttyán

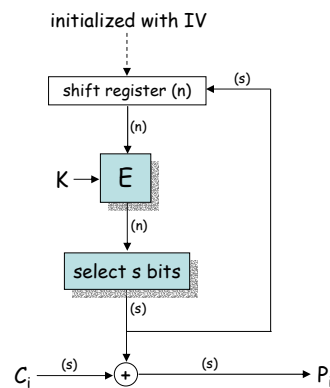
53

OFB mode

- encrypt



- decrypt



© Levente Buttyán

54

Properties of OFB mode

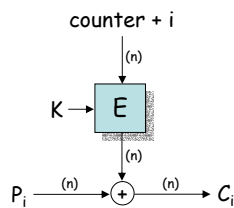
- a different IV should be used for every new message, otherwise messages will be encrypted with the same key stream
- the IV can be sent in clear
 - however, if the IV is modified by the attacker, then the cipher will never recover (unlike CFB)
- ciphertext block C_j depends on P_j only (does not depend on the preceding plaintext blocks)
 - however, rearranging ciphertext blocks affects decryption
- feedback size should be equal to n
 - ($= n$) \rightarrow cycle length is around 2^{n-1}
 - ($< n$) \rightarrow cycle length is around $2^{n/2}$
- error propagation:
 - one bit error in a ciphertext block C_j has an effect on the decryption of only that ciphertext block
 - P_j has bit errors where C_j had
 - an attacker may cause predictable bit changes in the j -th plaintext block
- error recovery:
 - recovers from bit errors
 - never recovers if bits are lost or the IV is modified

© Levente Buttyán

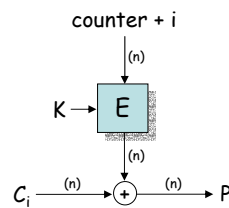
55

CTR mode

- encrypt



- decrypt



© Levente Buttyán

56

Properties of CTR mode

- similar to OFB
- cycle length depends on the size of the counter (typically 2^n)
- the i -th block can be decrypted independently of the others
 - parallelizable (unlike OFB)
 - random access
- the values to be XORed with the plaintext can be pre-computed
- at least as secure as the other modes

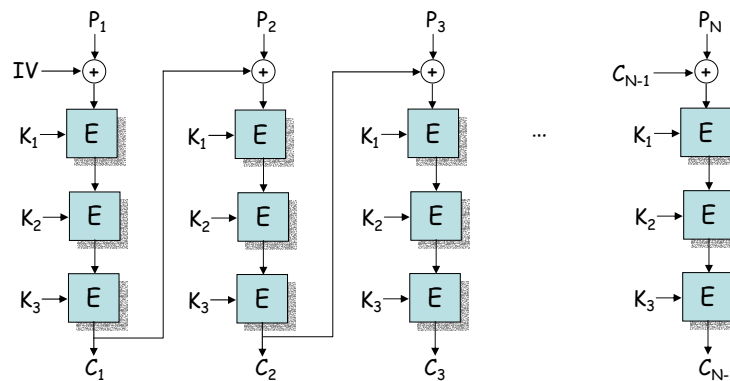
note1: in CFB, OFB, and CTR mode only the encryption algorithm is used (decryption is not needed)

- that is why Rijndael is optimized for encryption
- these modes shouldn't be used with public-key encryption algs.

note2: the OFB and CTR modes essentially make a synchronous stream cipher out of a block cipher, whereas the CFB mode converts a block cipher into a self-synchronizing stream-cipher

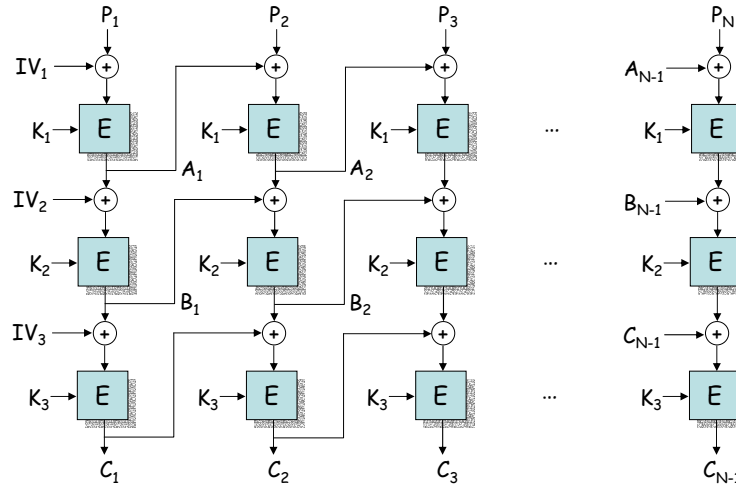
Multiple encryption modes of operation

- triple-outer-CBC



Multiple encryption modes of operation

- triple-inner-CBC

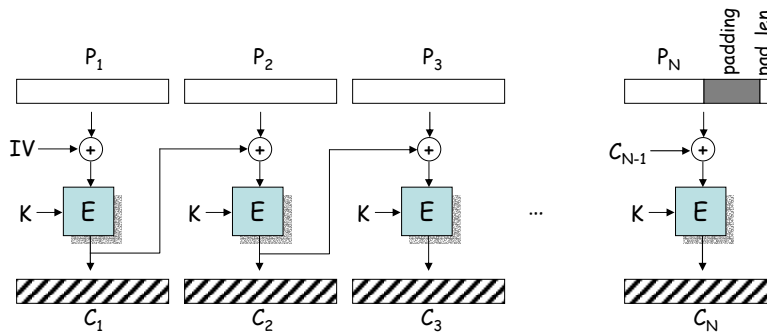


© Levente Buttyán

59

A security flaw induced by CBC padding

CBC encryption with padding



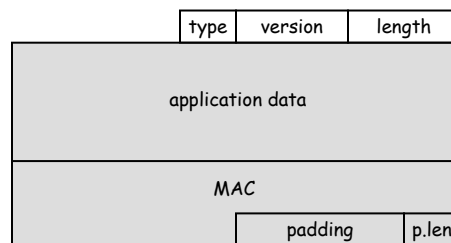
- TLS padding
 - last byte is the length n of the padding
 - all padding bytes have the value n
 - examples for correct padding: 0, 11, 222, 3333, ...
- verification of TLS padding:
 - if the last byte is n , then verify if the last $n+1$ bytes are all n

© Levente Buttyán

61

Side channel

- TLS record message format



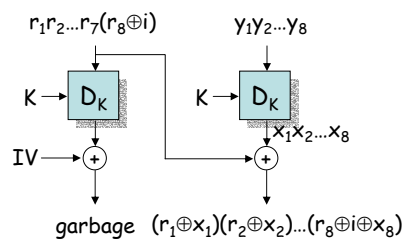
- send a random message to a TLS server
- the server will drop the message with overwhelming probability
 - either the padding is incorrect (the server responds with a DECRYPTION_FAILED alert)
 - or the MAC is incorrect with very high probability (the server responds with BAD_RECORD_MAC alert)
- if the response is BAD_RECORD_MAC, then the padding was correct
 - we get 1 bit of information!

© Levente Buttyán

62

Last byte(s) oracle

- assume we have an encrypted block $y_1y_2\dots y_8 = E_K(x_1x_2\dots x_8)$
- we want to compute x_8 (the last byte of x)
- idea:
 1. choose a random block $r_1r_2\dots r_8$; let $i = 0$
 2. send $r_1r_2\dots r_7(r_8 \oplus i)y_1y_2\dots y_8$ to the server (oracle)
 3. if there's a padding error, then increment i and go back to step 2
 4. if there's no padding error, then $r_8 \oplus x_8$ ends with 0 or 11 or 222 ...
 - the most likely is that $(r_8 \oplus i) \oplus x_8 = 0$, and hence $x_8 = r_8 \oplus i$



© Levente Buttyán

63

Last byte(s) oracle

- assume we get that $x \oplus r$ has a correct padding, but we don't know if it is 0 or 11 or 222 ...
- idea:
 1. let $j = 1$
 2. change r_j and send $r_1r_2\dots r_8y_1y_2\dots y_8$ to the server again
 3. if the padding is still correct then the j -th byte was not a padding byte; increment j and go back to step 2
 4. if the padding becomes incorrect then the j -th byte was the first padding byte; $x_j \oplus r_j | x_{j+1} \oplus r_{j+1} | \dots | x_8 \oplus r_8 = (8-j) | \dots | (8-j)$ and hence $x_j x_{j+1} \dots x_8 = r_j \oplus (8-j) | r_{j+1} \oplus (8-j) | \dots | r_8 \oplus (8-j)$

$x =$		DE AD BE EF DE AD BE EF			
$r =$		01 23 45 67 DD AE BD EC			
$r \oplus x =$		DF 8E FB 88 03 03 03 03			
j	r		$r \oplus x$		padding
1	00 23 45 67 DD AE BD EC	DE 8E FB 88 03 03 03 03			OK
2	00 22 45 67 DD AE BD EC	DE 8F FB 88 03 03 03 03			OK
3	00 22 44 67 DD AE BD EC	DE 8F FA 88 03 03 03 03			OK
4	00 22 44 66 DD AE BD EC	DE 8F FA 89 03 03 03 03			OK
5	00 22 44 66 DC AE BD EC	DE 8F FA 89 02 03 03 03			ERROR

$x_5 x_6 x_7 x_8 = DD \oplus 03 \text{ AE} \oplus 03 \text{ BD} \oplus 03 \text{ EC} \oplus 03 = \text{DE AD BE EF}$

© Levente Buttyán

64

Block decryption oracle

- assume we have an encrypted block $y_1y_2\dots y_8 = E_k(x_1x_2\dots x_8)$ and we know the value of $x_jx_{j+1}\dots x_8$ (using the last byte(s) oracle)
- we want to compute x_{j-1}
- idea:
 1. choose a random block $r_1r_2\dots r_8$ such that $r_j = x_j \oplus (9-j)$; $r_{j+1} = x_{j+1} \oplus (9-j)$; ... $r_8 = x_8 \oplus (9-j)$;
 2. let $i = 0$
 3. send $r_1r_2\dots r_{j-2}(r_{j-1} \oplus i)r_j\dots r_8y_1y_2\dots y_8$ to the server (oracle)
 4. if there's a padding error then increment i and go back to step 3
 5. if there's no padding error then $x_{j-1} \oplus r_{j-1} \oplus i = 9-j$ and hence $x_{j-1} = r_{j-1} \oplus i \oplus (9-j)$

```

x = DE AD BE EF DE AD BE EF
r = 01 23 45 67 DA A9 BA EB
r⊕x = DF 8E FB 88 04 04 04 04

i   r                               r⊕x                               padding
0   01 23 45 67 DA A9 BA EB      DF 8E FB 88 04 04 04 04      ERROR
1   01 23 45 66 DA A9 BA EB      DF 8E FB 89 04 04 04 04      ERROR
... ..
140 01 23 45 EB DA A9 BA EB      DF 8E FB 04 04 04 04 04      OK

x4 = 67⊕8C⊕04 = EF
    
```

© Levente Buttyán

Decryption oracle

- assume we have a CBC encrypted message (C_1, C_2, \dots, C_N) where
 - $C_1 = E_k(P_1 \oplus IV)$
 - $C_i = E_k(P_i \oplus C_{i-1})$ (for $1 < i < N$)
 - $C_N = E_k([P_N | \text{pad} | \text{plen}] \oplus C_{N-1})$
- we want to compute P_1, P_2, \dots, P_N
- idea:
 - decrypt C_N using the block decryption oracle and XOR the result to C_{N-1} ; you get $P_N | \text{pad} | \text{plen}$
 - decrypt C_i using the block decryption oracle and XOR the result to C_{i-1} ; you get P_i
 - decrypt C_1 using the block decryption oracle and XOR the result to IV; you get P_1 (if IV is secret you cannot get P_1)

complexity of the whole attack:

on average we need only $\frac{1}{2} * 256 * 8 * N = 1024 * N$ oracle calls !

© Levente Buttyán

Application

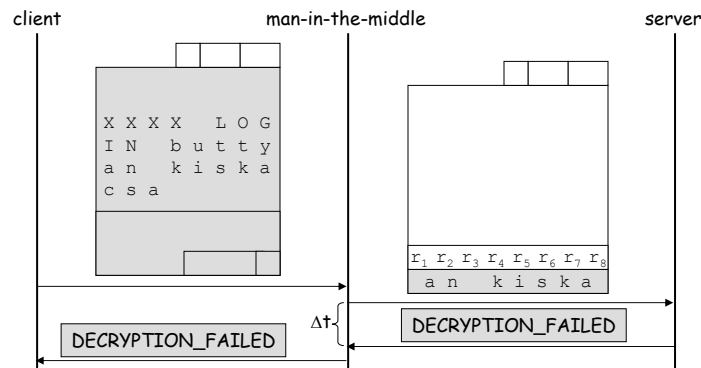
- vulnerable protocols: SSL, TLS, WTLS, IPsec, ...
- some notes:
 - problems (and solutions) with TLS
 - alert messages are encrypted → BAD_RECORD_MAC and DECRYPTION_FAILED cannot be distinguished
 - measure timing between oracle call and oracle response
 - BAD_RECORD_MAC takes more time than DECRYPTION_FAILED
 - BAD_RECORD_MAC and DECRYPTION_FAILED are fatal errors → connection is closed after one oracle call
 - a password can still be broken if it is sent periodically to a server using TLS (a different session is used each time the password is sent)

© Levente Buttyán

67

Example: IMAP over TLS

- Outlook Express checks for new mail on the server periodically (every 5 minutes)
- each time the same password is sent for every folder
XXXX LOGIN "username" "password"<OD><OA>
- it is possible to uncover the password using the attack as follows:



© Levente Buttyán

68

Fixes

- randomize response time after an error occurred (measuring timing of alert messages won't work)
- use random padding bytes
- put the padding before the MAC !

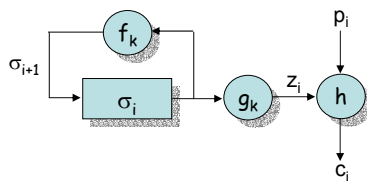
Stream ciphers

- general principles
- one-time pad
- LFSR based stream ciphers
- RC4

General principles

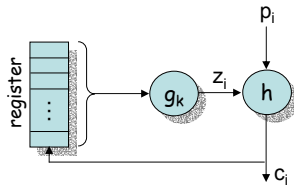
- while block ciphers simultaneously encrypt groups of characters, stream ciphers encrypt individual characters
 - may be better suited for real time applications
- stream ciphers are usually faster than block ciphers in hardware (but not necessarily in software)
- limited or no error propagation
 - may be advantageous when transmission errors are probable
- note: the distinction between stream ciphers and block ciphers is not definitive
 - stream ciphers can be built out of block ciphers using CFB, OFB, or CTR modes
 - a block cipher in ECB or CBC mode can be viewed as a stream cipher that operates on large characters

Synchronous stream ciphers



- the key stream is generated independently of the plaintext and of the ciphertext
- properties:
 - needs synchronization between the sender and the receiver
 - if a character is inserted into or deleted from the ciphertext stream then synchronization is lost and the plaintext cannot be recovered
 - additional techniques must be used to recover from loss of synch.
 - insertion and deletion are easy to detect by the receiver
 - no error propagation
 - a ciphertext character that is modified during transmission affects only the decryption of that character
 - an attacker can make changes to selected ciphertext characters and know exactly what effect these changes have on the plaintext (if $h = \text{XOR}$)

Self-synchronizing stream ciphers



- the key stream is generated as a function of a fixed number of previous ciphertext characters
- properties:
 - self-synchronizing
 - since the size t of the register is fixed, a lost ciphertext character affects only the decryption of the next t ciphertext characters
 - more difficult to detect insertion and deletion of ciphertext char's
 - limited error propagation
 - if a ciphertext character is modified, then decryption of the next t ciphertext characters may be incorrect
 - modifications are easier to detect than in case of synch. stream ciphers
 - ciphertext characters depend on all previous plaintext characters
 - better diffusion of plaintext statistics

© Levente Buttyán

73

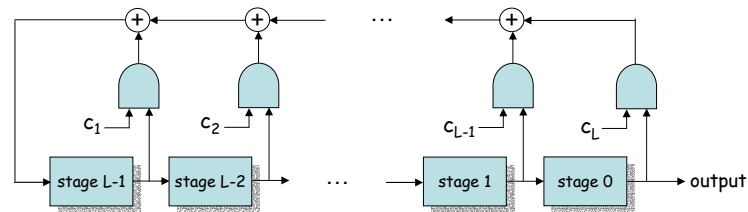
Vernam cipher and the one-time pad

- Vernam cipher
 - $c_i = p_i \oplus k_i$ for $i = 1, 2, \dots$
 - where p_i are the plaintext digits, k_i are the key stream digits, c_i are the ciphertext digits, and \oplus is the bitwise XOR operation
- one-time pad
 - a Vernam cipher where the key stream digits are generated independently and uniformly at random
 - the one-time pad is unconditionally secure [Shannon, 1949]
 - $I(P; C) = H(P) - H(P|C) = 0$
 - a necessary condition for a symmetric key cipher to be unconditionally secure is that $H(K) \geq H(P)$ [Shannon, 1949]
 - practically, the key must have as many bits as the compressed plaintext
 - impractical because of key management problems

© Levente Buttyán

74

Linear feedback shift registers (LFSR)



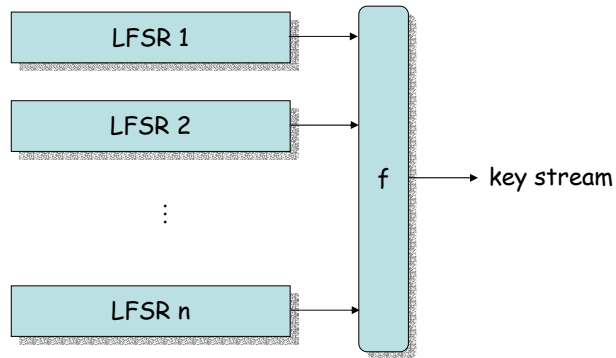
- where each $c_i \in \{0, 1\}$ and each stage can store 1 bit
- operation is controlled by a clock
- during each time unit
 - the content of stage 0 is output
 - the contents of a fixed subset of stages are XORed
 - the content of stage i is moved to stage $i-1$ ($i = 1, 2, \dots, L-1$) and the result of the XOR operation is moved in stage $L-1$ (feedback)

Maximum length LFSR and linear complexity

- maximum length LFSR
 - since the number of possible states of an LFSR is finite, the output stream it produces is always periodic
 - an LFSR is maximum length LFSR, if it produces an output sequence with maximum possible period 2^L-1 (m-sequence)
 - m-sequences have good statistical properties
- linear complexity
 - the linear complexity $L(s^n)$ of a finite binary sequence s^n is the length of the shortest LFSR that generates a sequence having s^n as its first n elements
 - Berlekamp-Massey algorithm:
 - let s be a binary sequence of linear complexity L , and let t be a finite subsequence of s of length at least $2L$
 - the Berlekamp-Massey algorithm with input t determines an LFSR of length L that generates s
 - an LFSR should not be used as a key stream generator, as the resulting stream cipher would be vulnerable to known-plaintext attacks

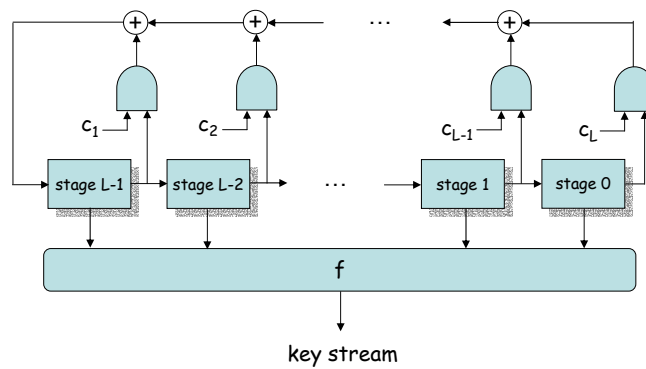
Stream ciphers based on LFSRs

- nonlinear combination generators



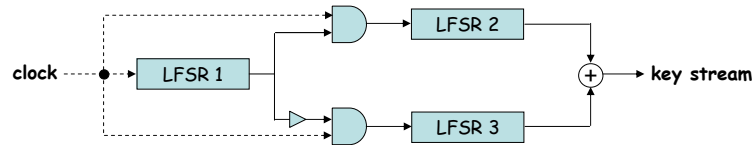
Stream ciphers based on LFSRs

- nonlinear filter generators

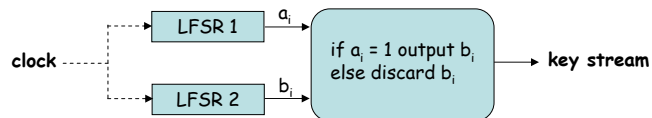


Stream ciphers based on LFSRs

- clock-controlled generators
 - alternating step generator



- shrinking generator



© Levente Buttyán

79

RC4 stream cipher

- initialization (input: a seed K of keylen bytes)


```
for i = 0 to 255 do
    S[i] = i;
    T[i] = K[i mod keylen];
```
- initial permutation


```
j = 0;
for i = 0 to 255 do
    j = (j + S[i] + T[i]) mod 256;
    swap(S[i], S[j]);
```
- stream generation (output: a stream of pseudo-random bytes)


```
i, j = 0;
while true
    i = (i + 1) mod 256;
    j = (j + S[i]) mod 256;
    swap(S[i], S[j]);
    output S[(S[i] + S[j]) mod 256];
```

© Levente Buttyán

80