CRYPTOGRAPHY LECTURE, 2022

Computer Science and Algorithmics, PWr

Mirosław Kutyłowski

Symmetric encryption

"symmetric"

- the same key used for encryption and decryption
- two subcategories:
  - stream ciphers (already discussed)
  - block ciphers – $\mathrm{Enc}\colon \mathrm{Keyspace} \times \{0,1\}^k \rightarrow \{0,1\}^m$

    (typically $k = m$ and $k = 128, 256, 512, \ldots$ but not much higher

CPA

Chosen plaintext attack (CPA):

- adversary can ask for pairs (plaintext,ciphertext)
- **non-adaptive CPA:** the pairs are given before analysis starts
  - the case when an attacker knows the messages encrypted (e.g. if stolen secret documents are leaked to a spy)
- **adaptive CPA:** the attacker can ask for a ciphertext of any plaintext
  - the case when an attacker holds a tamper-proof device and wishes to learn the key stored inside

KPA

Known plaintext attack (KPA):

- adversary has pairs (plaintext,ciphertext) but cannot choose them

Usually situation harder for breaking a cipher:

i. impossible to use plaintexts with some dependencies

ii. so e.g. *differential cryptanalysis* does not apply

Key length, CPA, KPA and brute force attack:

- try all keys:
  - for a key K and pair (P, C) of (plaintext,ciphertext) test if
  $$\mathrm{Enc}_K(P) = C?$$
  - if $\neq$, then K is wrong
  - if $=$, then with a high probability K is correct
- if the key has 128 bits, then at average 2^{127} tests needed
- a year $\approx 2^{25}$ seconds, if 1 million tests per second, then $\approx 2^{45}$ tests per year
- use 1 million computers and 1024 years: 2^{75} tests
  - probability to success 2^{-53} (practically =0)

Conclusion: brute force attack does not work for any reasonable key size provided that the keys are random

Key length and brute force attack

- the conclusion does not apply to the case where the key space is small:
  - memorizable passwords
  - long sentences in natural languages (e.g. parts of "Pan Tadeusz")
  - anything one can guess

Case: WIFI passwords (usually not a string of random 80 characters)

Ciphertext-only attack

the attacker knows only a set of ciphertexts.

Test: trial decrypt and look if the obtained plaintext makes sense (a message in a natural language)

The test does not work if the plaintext is a random string (e.g. a key)

Key is not the only target!

- some possible plaintext
  - e.g.:

    $P_1 =$"concentration point for the aircraft carriers is north of Midway"
    or

    $P_2 =$"concentration point for the aircraft carriers is south of Midway"
- or US Navy it was enough to learn whether C encrypts P_1 or P_2 , it is not necessary to learn Japanese secret key used

Semantic security

given plaintexts P_1, P_2, a ciphertext C encrypting P_b where b is chosen at random, then

it is infeasible to learn b with non-negligible advantage

Double encryption - warning

an idea to increase the key size: encrypt twice with different keys

$$\mathrm{Enc}_{K,K'}(M) = \mathrm{Enc}_K(\mathrm{Enc}_{K'}(M))$$

brute force seems to be much harder (guess twice as many bits!) but this is not

Attack based on birthday paradox

Triple DES

$$\mathrm{Enc}_{K,K'}(M) = \mathrm{Enc}_K(\mathrm{Dec}_{K'}(\mathrm{Enc}_K(M)))$$

– if $K = K'$ then it reduces to DES (backwards compatibility)

– birthday attack does not work

Avelanche effect (efekt lawinowy)

If the keys K_1 and K_2 are somehow related (e.g. differ by just 1 bit), then it is infeasible to guess any relationship between $\mathrm{Enc}_{K_1}(M)$ and $\mathrm{Enc}_{K_2}(M)$

Otherwise: it would be easier to break codes like in the movies

Block ciphers

– each plaintext is a block of a fixed length

– $\mathrm{Enc}\colon \{0,1\}^k \times \{0,1\}^m \rightarrow \{0,1\}^m$

Notes:

1) the ciphertext cannot be shorter than the plaintext - Shannon's theorem

2) ciphertext longer than a plaintext might be a problem for practical reasons

$\longrightarrow$ encrypting a whole disk would require moving to a larger disk !

$\longrightarrow$ problem for operating system (pagesize, . . .)

Choice of block size m

- very small m problematic: frequence analysis attack
- large m problematic: difficulty to run encryption/decryption efficiently on weak machines
- compromise: AES: $m = 128$, its proposal (Rijndael): $m = 128, 192, 256$

AES competition (Advanced Encryption Standard)

- run by NIST – a US authority
- open competition
- narrowing the list of candidates, workshops, call for comments, attacks, …
- public set of requirements

Some requirements

platform independent:

- efficient both on special hardware and general purpose computers
- open for different implementation strategies:
  - via (complex) algebraic operations
  - with lookup tables

- former approach (DES - Data Encryption Standard from 1975): design it so that
  - a software implementation should be very slow
  - a hardware implementation is very fast

Some requirements

transparency:

- no mysterious components, no security by obscurity
- whitepaper MUST explain the construction

Rounds

idea: repeat the same operations – e.g. 10 identical rounds (with different data)

- easier to implement (codesize!, hardware size!)
- easier security analysis

Key Schedule

for each round a different **subkey**

- subkeys derived from the main key via some algorithm (key schedule)
- subkeys should be “independent” and not ease cryptoanalysis

AES - construction of Rijndael

- working on a table of 4×4 table of bytes
- 10 rounds (initial round and the last round are slightly different)
- a round consists of 4 parts:
  - $\longrightarrow$ SubBytes – a non-linear substitution according to a lookup table (S-Box)
  - $\longrightarrow$ ShiftRows – cyclic shift on rows: 0-shift on row 0, 1-shift on row 1, 2-shift on row 2, 3-shift on row 3
  - $\longrightarrow$ MixColumns – a linear operation on each column
  - $\longrightarrow$ AddRoundKey: xor with round subkey

AES details

MixColumns operation:

linear algebra view:

$$\begin{bmatrix} b_{0,j} \\ b_{1,j} \\ b_{2,j} \\ b_{3,j} \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} a_{0,j} \\ a_{1,j} \\ a_{2,j} \\ a_{3,j} \end{bmatrix} 0 \le j \le 3$$

or as polynomials over $\mathrm{GF}(2^8)$:

polynomial $b(x) = b_3x^3 + b_2x^2 + b_1x + b_0$ multiplied by $a(x) = 3x^3 + x^2 + x + 2$

modulo $x^4 + 1$

Alternative implementation with lookup tables

$$\left(x_0^{(r+1)}, x_1^{(r+1)}, x_2^{(r+1)}, x_3^{(r+1)}\right) := T_0(x_0^r) \oplus T_1(x_5^r) \oplus T_2(x_{10}^r) \oplus T_3(x_{15}^r) \oplus K_0^{(r+1)}$$

$$\left(x_4^{(r+1)}, x_5^{(r+1)}, x_6^{(r+1)}, x_7^{(r+1)}\right) := T_0(x_4^r) \oplus T_1(x_9^r) \oplus T_2(x_{14}^r) \oplus T_3(x_3^r) \oplus K_1^{(r+1)}$$

$$\left(x_8^{(r+1)}, x_9^{(r+1)}, x_{10}^{(r+1)}, x_{11}^{(r+1)}\right) := T_0(x_8^r) \oplus T_1(x_{13}^r) \oplus T_2(x_2^r) \oplus T_3(x_7^r) \oplus K_2^{(r+1)}$$

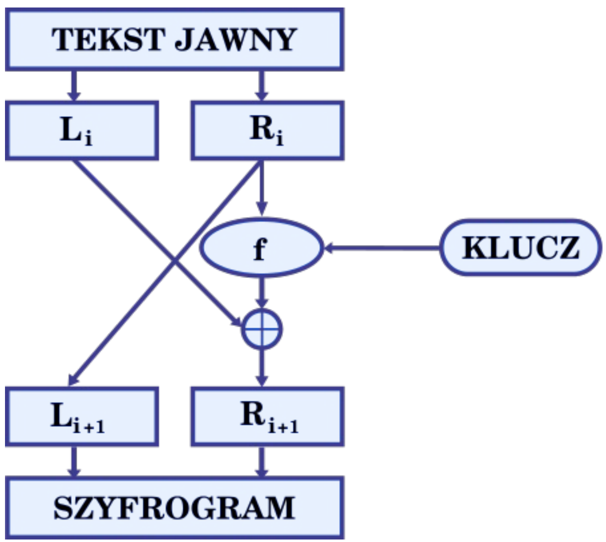
$$\left(x_{12}^{(r+1)}, x_{13}^{(r+1)}, x_{14}^{(r+1)}, x_{15}^{(r+1)}\right) := T_0(x_{12}^r) \oplus T_1(x_1^r) \oplus T_2(x_6^r) \oplus T_3(x_{11}^r) \oplus K_3^{(r+1)}$$

Feistel construction

– round organization so that it is "one-way" but stil invertible

– many symmetric encryption schemes follow this trick

Diagram (from Wikipedia, ignore the part tekst jawny and szyfrogram):



Feistel construction:

round i: intermediate state (L_i, R_i)

encryption in round $i+1$:

$$L_{i+1} = R_i,$$

$$R_{i+1} = L_i \otimes F(R_i, K_i),$$

decryption in the reverse order of rounds:

$$R_i = L_{i+1}$$

$$L_i = R_{i+1} \otimes F(L_{i+1}, K_i).$$

Standard Attack Methods

1. differential cryptanalysis
2. linear cryptanalysis
3. side channel leakage
4. fault cryptanalysis

Differential cryptanalysis

- S-boxes is a source of problem:
  - inputs: X and $X \otimes \Delta$ (difference Δ)
  - after adding a round subkey: $X \otimes K$, $X \otimes \Delta \otimes K$
  - adding the round key K does not change the difference
  - what is the difference after applying the S-Box?
  - precomputed table:

    $$A, A \otimes \Delta \to A', A' \otimes \Delta_1$$

    $$B, B \otimes \Delta \to B', B' \otimes \Delta_2$$

    $$\ldots$$

  - observe output difference. If it is, say, Δ_2, then we may assume that $X \otimes K = B$, or $X \otimes K = B \otimes \Delta$ and solve for K

... not that easy

because we do not know the output of the S-Box

Construction of *characteristics*:

1. assuming that at a given place S the difference is Δ' estimate probabilities that in the output the difference at a certain point D is Γ
2. if the assumption about the difference at S was false, then assume that at D the differences are totally random

in case 1 the differences are usually very small, ... but with a large number of pairs of inputs X, $X \otimes \Delta$ one can see some statistical bias indicating true value of Δ'

Finding effective charateristics: very complex task

Linear cryptanalysis

non-linear operations (S-Boxes) approximated by linear operations

- **find linear equations in input bits, output bits and key bits that are true for more than 50% of cases**
- move keybits to one side and other bits to the other side -- thus get an expression for keybits
- advantage should be statistically observable

Toy Example

$i_{17} \otimes i_{64} \otimes k_{23} = o_{22} \otimes 1$ that is true with probability 0.50000123483029

yields an expression:

$k_{23} = o_{22} \otimes 1 \otimes i_{17} \otimes i_{64}$

that is more likely to be true than false

gather statistics, after some number of samples there will be a strong bias towards the true value

in practice we have to combine the expressions from all rounds

Side channel leakage

leakage via operations executed.

Power analysis:

- executing $x := x \otimes k$ means swapping the bit x if $k = 1$ and no change otherwise
- changing the state of a 1-bit memory costs more energy than keeping the old value

so observe that power consumption to learn k

problems in practice:

i. noise

ii. sampling power traces

iii. finding the right moment

Fault attacks

1. encrypt M with (hidden) key K
2. encrypt M with K again, but with a laser set one bit register A to 1
3. compare the results. If unequal, then originally A contained a 0

example attack point:

an input bit XOR-ed with the key bit during the last round of AES

Encryption modes for block encryption schemes

– what if the plaintext is not a single block as described for Enc?

– needed: encryption modes that enable to encrypt a file of any length

Padding:

padd the plaintext so that it can be divided to some number of full blocks:

i.e. $128 \cdot n$ for AES with 128 bit blocks

padding must be reversible

(e.g. the last byte in the block must say how many bits have been added)

Initial vector

- many modes require IV – the initial vector
- thumb rule: never repeat the same IV with the same key
- use for instance: current time + counter value
- IV transmitted in clear!

Electronic Codebook (ECB)

- $C_i = \mathrm{Enc}_K(P_i)$
- advantages:
  - simple
  - modification of a single block of plaintext $\Rightarrow$ modification of one block of ciphertext
- deadly threats:
  - if $P_i = P_j$ then $C_i = C_j$. This leaks a lot of information!

Cipher Block Chaining (CBC)

encryption:

$$C_{i+1} = \mathrm{Enc}_K(C_i \otimes P_{i+1}),$$

$$C_0 = \mathrm{IV}$$

decryption:

$$P_{i+1} = \mathrm{Dec}_K(C_{i+1}) \otimes C_i$$

advantages:

- $P_i = P_j$ does not imply that $C_i = C_j$
- C_i depends on $P_1, \ldots, P_i$
- manipulation on one ciphertext block destroys all remaining plaintext blocks

disadvantages:

- replacing a single plaintext block requires re-encryption starting at this block

  (think about disk encryption!)

Cipher Feedback mode (CFB)

Encryption:

$C_0 = \mathrm{IV}$

$C_{i+1} = \mathrm{Enc}_K(C_i) \otimes P_{i+1}$

decryption:

$P_{i+1} = C_{i+1} \otimes \mathrm{Dec}_K(C_i)$

Advantages:

- C_i depends on all $P_1, \ldots, P_i$
- some advantage with encryption rate if P_i's come irregularly

Counter mode (CTR)

encryption

$Y_i = \mathrm{Enc}_K(\mathrm{IV} + f(i))$, where f is some counter function

$C_i = Y_i \otimes P_i$

decryption

$Y_i = \mathrm{Enc}_K(\mathrm{IV} + f(i))$,

$P_i = Y_i \otimes C_i$

(dis)advantages:

– in order to replace P_i by P_i' it suffices to compute

  $C_i := C_i \otimes (P_i \otimes P_i')$

– ... so it is better to use authenticated CTR mode

  (ps: use CCM and not GCM! if you need explanation enroll to Security& Cryptography Master program)

Authenticated block encryption mode

change the encryption scheme so that:

- a manipulation of a ciphertext results in an invalid ciphertext with very high probability

AES+ CBC

a change in one block results in changes in two (unpredictable) changes in two consecutive plaintext blocks

(recall that $P_{i+1} = \mathrm{Dec}_K(C_{i+1}) \otimes C_i$)

$\longrightarrow$ usually after such manipulation we get 2 blocks of random nonsense

$\longrightarrow$ ... but if the plaintext was a random string, then one cannot detect the manipulation

Format Preserving Encryption

- standard block ciphers have certain minimal length: `blocksize=128`, ...

- encrypting n-bit plaintext to n-bit ciphertexts is not a problem if $n \geq \mathrm{blocksize}$

- the challenge is: how to convert k-bit strings into k-bit ciphertexts for small k?

  – application: encrypting credit card number in a record of the same size in a database

Challenges:

i scaling down constructions such as AES is not a good idea: their security argument depend heavily on the necessary block size

ii if an attack requires $2^{\text{blocksize}/2}$ steps, then it is ok for AES (2^{64} steps is a huge number) while for $\text{blocksize} = 24$, making 2^{12} steps is a negligible effort

Solution ideas:

idea 1: m bit plaintext X, n-bit block cipher Enc with key K

i. $C := 0^{n-m} || X$

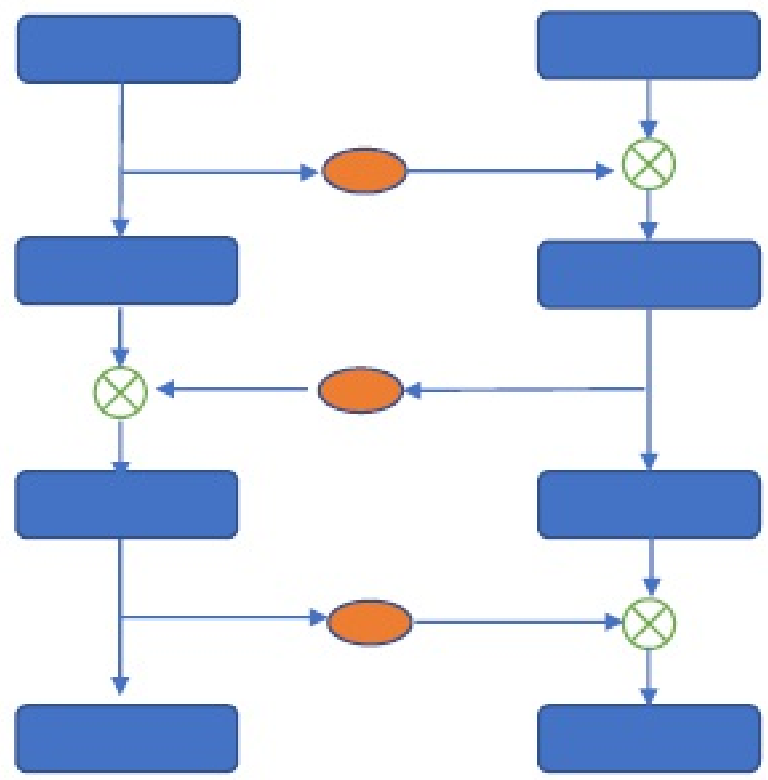
ii. $C := \mathrm{Enc}_K(C)$

iii. if C starts with $n-m$ zeroes, then output Cafter truncating these zeroes, else goto ii

Decryption (obvious)

disadvantage: encryption and decryption times are random variables, computationally intensive

Feistel constructions



the orange circle stands for, say AES for input padded with m zeroes, while in the output m bits are truncated

NIST Standards

FF1, FF3, (FF2 broken during the standardization process)

- number of rounds below what suggested by cryptographers
- security broken if you may query some number of (plaintext, ciphertext) pairs
- some design decisions not compliant with the state-of -the-art

nevertheless offered and used