BPSI 2/10

WTOREK, B2-6
12-14

maciejg@amu.edu.pl

[I] + [II] = 100 pkt <=> OCENA Z EGZAMINU
w okolicach grudnia — SESJA

* W. Stallings - Cryptography and Network Security
* M. Kutyłowski - Kryptografia, Teoria, ...
* B. Schneier, Kryptografia

Kryptologia → KRYPTOGRAFIA, KRYPTOANALIZA

KRYPTOGRAFIA
* projektowaniem algorytmów kryptograf.
* protokołów kryptograficznych

KRYPTOANALIZA
* łamaniem (znajdowanie słabych punktów)

NIGDY NIE MAMY PEWNOŚCI PRZY KRYPTOANALIZIE. SĄ W UKRYCIU.

Protokół kryptograficzny - biorą udział w nim conajmniej dwie strony (Alice i Bob), ale większej np. (Alice, Celine, Dave, Bob)

A B C D

EVE - pasywny napastnik (może podsłuchiwać ...)

MALLET - aktywny napastnik (może podmieniać, zamieniać pakiety ...)

OCZYWIŚCIE PRZY DOWOLNYM SCENARIUSZU

M - wiadomość (message) skończony ciąg bitów

E - algorytm szyfrujący [E, D]

D - algorytm deszyfrujący

K - klucz (losowy ciąg bitów) może być do szyfrowania albo deszyfrowania albo do tego i do tego.
bywa tajny, czyto

C - szyfrogram (zaszyfrowana wiadomość)

ENIGMA ? lata '70

Algorytm (RC4) - '90 szyfrujący

CHECK FINGERPRINT

$E_k(M) = C$

$D_{k'}(C) = M$

SYMETRYCZNY PROTOKÓŁ SZYFROWANIA

CEL: Poufne przekazywanie informacji

algorytm szyf, deszyf.

M [E,D]K ALICE — [E,D]K BOB

1) Jawnie Alice i Bob wybierają [E,D] (sposób szyfr.) na otwartym kanale. Oboje muszą być świadomi o bezpieczeństwie

2) „Bezpiecznie" – nikt poza nimi nie wie/nie zna K ani [E,D] (tajny)

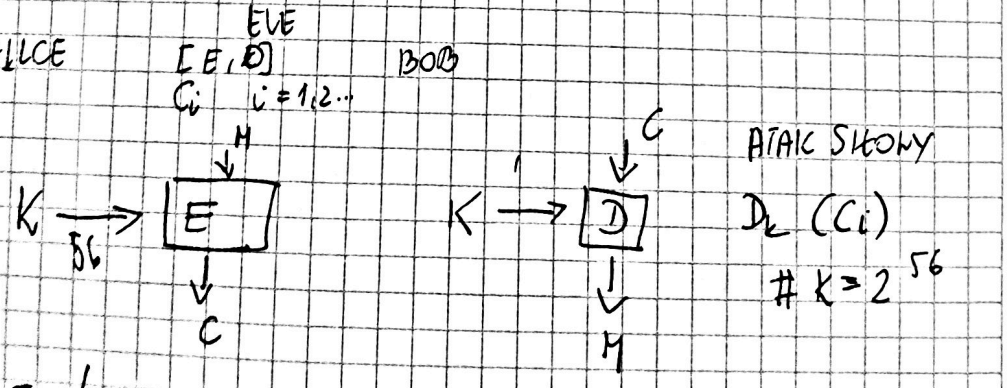
3) Alice szyfruje M tzn $C = E_K(M)$

4) Alice C do Boba

5) Bob wymianę z K lub do deszyfr. K'

6) Bob deszyfruje C tzn $M = D_{k'}(C)$

Symetryczny, bo z K łatwo możemy wyznać K'



$E \in$ {DES, 3DES, IDEA, AES, RC5}

DES: $|K| = 56$, $|M_i| = 64$; 3DES: $|K| \in \{112, 168\}$, $|M_i| = 64$; AES, RC5: $K \in \{128, 196, 256\}$, $M_i = \{128, 196, 256\}$

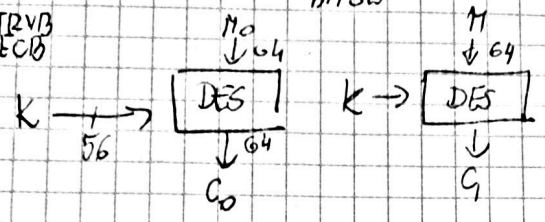
3 dni → 19 h

Udowodniono, że nie jest doskonała

SZYFRY

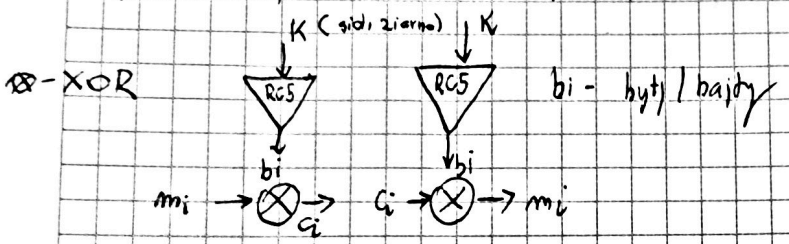
→ SZYFRY BLOKOWE

$M = M_0 M_1 \dots M_s$ | M_i - blok o ustalonej l. bitów



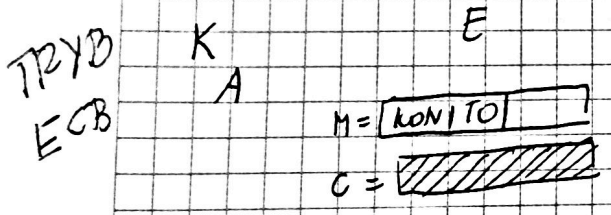
W TYM PRZYP. SZYFROGRAM MOŻE BYĆ DŁUŻSZY OD WIADOMOŚCI.

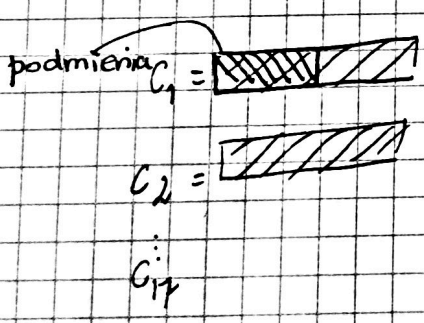
→ SZYFRY STRUMIENIOWY



⊗	1	0
0	1	0
1	0	1

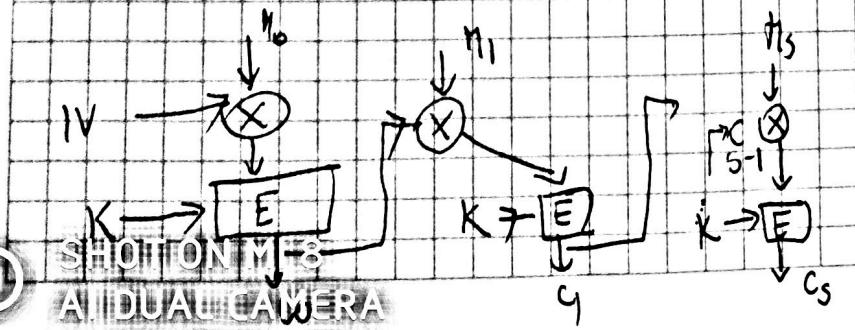
TRYB ECB

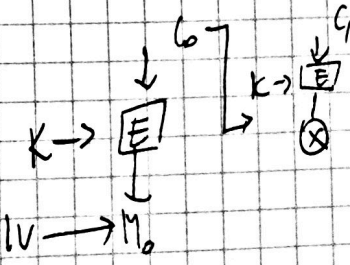




TRYB CBC

IV - Initial Value (jawna)





ONE TIME PAD

M = 1101

K = 1010 LOSOWY

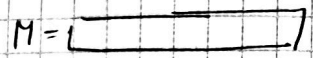
E C = M XOR K

M 1101
K 1010 ← losowy
C = 0110

D = M = C XOR K

C = 0110
K = 1010
M = 1101

mi jesteśmy w stanie

M = 

Mi = 

$c_i \otimes c_{i+j} = (k_i \otimes m_i) \otimes (k_{i+i} \otimes m_{i+j}) =$

$= 0 \otimes m_i \otimes m_{i+j}$

BPSI - wykład 2 (10/10/2019)

$H: M \to S$

M – zbiór wiadomości

S – zbiór wartości H (zbiór skrótów, hash)

H spełnia własności:

Jeśli jednocześnie spełnia te dwa warunki to mówimy, że H jest jednokierunkowa

1) dla dowolnego M (wiadomości) $M \in \mathcal{M}$ obliczenie $H(M)$ jest łatwe
2) dla danego $s \in S$ obliczenie $H(S)^{-1}$ jest trudne

3) długość bitowa skrótów (hashu) jest ustalona i mniejsza od argumentu funkcji H.

RIMPED

$H(M) = s$ – długa bitowa ustalona

SHA1
SHA2 ⋮
SHA3

Def Znalezienie dwóch wiadomości M_1 i M_2 takich, że $M_1 \neq M_2$, $H(M_1) = H(M_2)$ nazywamy konfliktem / kolizją dla H

Funkcja H jest bezpieczna jest wtedy gdy nie da się obliczyć / znalezienie konfliktu jest obliczeniowo niemożliwe nietrudne")

Przykłady / Zadania:

$H(M) = S$, S ustalone

1) Dane: s – hash, H – funkcja haszująca, M – wiadomość
Problem $M' \neq M$ takie, że $H(M') = H(M)$

Funkcja jest bezkonfliktowa jeśli P_1 jest obliczeniowo mały.

2) Dane H – hash

Problem 2. Znajdź M_1, M_2, $M_1 \neq M_2$ takie, że $H(M_1) = H(M_2)$

H jest silnie bezkonfliktowa jeśli Problem 2 jest „trudny"

3) Dane jest K, M, S ~~$H(M)=S$~~

Problem 3. znaleźć M takie, że $H(M) = S$

H jest jednokierunkowa jeśli Problem 3 jest „trudny"

TW. Niech $K: X \to Z$ będzie funkcją skrótu (kompresyjną $|X| \geq 2|Z|$) załóżmy że A jest algorytmem odwracającym K który istnieje algorytm probabilistyczny, czyli dla Vegas) który dla K znajduje kolizję z prawdopodobieństwem $\frac{1}{2}$.

Algorytm B:

1) Losuje $x \in X$

2) oblicza $z = H(x)$

3) oblicza $x_1 = A(z)$

4) Jeśli $x_1 \neq x$ to wynik (x_1, x) (konflikt dla H)

w.w.p.

4) Klucz: k $\quad S = \frac{1}{N\text{-bitów}}$

PARADOKS URODZINOWY

M_1 $\quad H(M_1) = S_1$ $\quad S_1 = S_2$?

M_2 $\quad H(M_2) = S_2$ $\quad S_3 = S_1 \vee S_3 = S_2$

M_3 $\quad H(M_3) = S_3$ $\quad x \approx C_1 \, 2^{\frac{N}{2}}$

$\vdots \; M_k$ $\quad p = \frac{1}{2}$ $\quad C_1 = 1{,}17$

M_{2^N+1} $\quad H(M_{2^N+1}) = S_{2^N+1} \Rightarrow P(H) = 1$

ZOBOWIĄZANIE BITOWE - PROTOKÓŁ

- kryptografia symetryczna $\quad b \in \{0,1\}$ - decyzja bitowa

$[E, D]$

BOB $\quad$ PODJĘCIE DECYZJI PRZEZ $\quad$ ELICE

1) losuje k-ciąg bitów

2) $\xrightarrow{R}$ (NIE MOŻEMY 1b) ZMIENIĆ)

3) Alice podejmuje decyzję b

4) klucz tajny do E

5) oblicza $C = E_k(Rb)$ ← zaszyfrowana bitu

$\xleftarrow{C}$ 6)

UJAWNIENIE DECYZJI ALICE

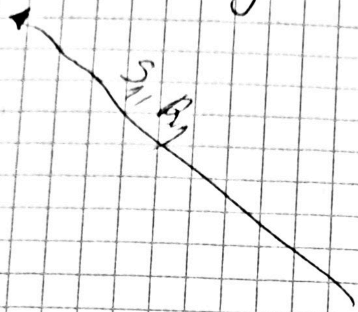
BOB (jest przekonany, że Alice nie zmieniła decyzji) $\xleftarrow{}$ 1) ALICE

2) Obliczanie $R'b = D_k(C)$

3) Jeśli $R = R'$ to Bob akceptuje decyzję Alice

• Używamy f. haszujące

Bob — PODEJMOWANIE DECYZJI — Alice



1) R_1, R_2

2) Ustala decyzję $b \in \{0,1\}$

3) oblicza $S = H(R_1 R_2 b)$ — losowo generowane

4)

$H(R_1 \_\_\_) = S$

UJAWNIENIE DECYZJI

$H(R_1 R_2' 1-b) = S$ — ZMIANA DECYZJI

BOB — $R_1' = R_1$ lub $R_1' \neq R_1$ — ALICE

1) R_1', R_2, b →

2) oblicza $H(R_1' R_2 b) = S'$

3) po jeśli $S' = S$ i $R_1' = R_1$

to Bob uznaje decyzję b Alice

RZUT MONETĄ

ALICE — Bob

1) zobowiązuje się do b $\in \{0,1\}$ wst. zobowiązanie bitowe

2) losuje bit $c \in \{0,1\}$

3) wysyła go ALICE

4) odkrywa decyzję Boba

5) wynik rzutu do $w = c$ XOR b

TRENT S - sekret

1) losuje R
2) oblicza L = S XOR R
3) L do Alice, R do Boba

Aby uzyskać sekret S Alice i Bob wspólnie obliczają
S = L XOR R

Dzielenie sekretu

S- sekret

PODZIAŁ
1) losuje R
2) oblicza L = S xor R
3) przesyła L Alicji, R Bobowi

ALICJA
BOB

ODZYSK INFORMACJI
1) Alicja i Bob wspólnie obliczają
S = L xor R

Podział
1) losujemy $R_1 \dots R_n$
2) Obliczamy $L = R_1$ xor $R_2 \dots$ xor S
3) Dzielimy R_i $i = 1, n+1$ stronami t_i

1) A_i $i = 1, n+1$ się wspólnie obliczają

$$S = R_1 \text{ xor} \dots \overset{\text{xor}}{\underset{n+1}{}}$$

Dzielenie tajemnicy z progiem

1) S- sekret zostaje rozdzielony między n osobami
2) Każda grupa osób liczona z $t \leq n$ osób może wyznaczyć S
3) Każda grupa liczna z mniej niż t osób nie jest wstanie obliczyć S.

Gdzie t nazywamy wartością progową.

INTERPOLACJA LAGRANGE'A

Dane (x_i, y_i) ; $i \leq 1 \leq t$ punkty płaszczyzny, przez które przechodzi wielomian $f(x)$, stopnia $(f) \leq t$

Wynika $f(x) = \sum_{i=1}^{t} y_i \prod_{\substack{1 \leq j \leq t \\ i \neq j}} \frac{x - x_j}{x_i - x_j}$

PROTOKÓŁ SHAMIRA

Rozdzielenie udziałów, S-sekret, n - liczba uczestników protokołu
t - wartość progowa

1) Losujemy liczbę pierwszą $p > \max(S, n)$
2) $a_0 = S$
3) Losujemy $a_1, a_2, \dots a_{t-1}$ tak, że $a_i < p$, $i = 1, t-1$
4) Definiujemy $f(x) = \sum_{i=0}^{t-1} a_i x^i$ ~~$\sum_{i=0}^{t} a_i x^i$~~ $\sum_{i=0}^{t-1} a_i x^i$
5) Obliczamy S_i (udziały) $S_i = f(i) \pmod p$
6) Przesyłamy „bezpiecznie" S_i do uczestników A_i ; $i = 1 \dots n$

odzyskanie S

Dowolna grupa t uczestników A_j $j = 1 \dots t$ wspólnie odzyskują S metodą Lagranga dodatkowo chiełpc modulo p.

PRZYKŁAD

$p = 13 \quad s = 4 \quad t = 3 \quad n = 5$

1) $p = 13$

2) $a_0 = 4$

3) ~~$(a_1 = 4 \quad a_2 = 7 \quad a_3 = 10) \leftarrow 13$~~ 3) $a_1 = 7, \; a_2 = 10$

4) ~~$f(x) = 4 + 4x + 7x^2 + 10x^3$~~ 4) $4 + 7x + 10x^2 = f(x)$

5) $s_1 = f(1) \equiv 8 \mod 13$

$s_2 = f(2) \equiv 6 \mod 13$

$s_3 = f(3) \equiv 11 \mod 13$

$s_4 = 10 \mod 13$

$s_5 = 3 \mod 13$

$A_1 \quad (A_2) \quad A_3 \quad (A_4) \quad (A_5)$ $\qquad 3^{-1} = 9 (13)$

$(1, 8)$ (x_1, y_1); $(2, 6)$ (x_2, y_2); $(4, 10)$ (x_3, y_3); $(5, 3)$

WYBIERAMY (ONI WSPÓLNIE ODZYSKUJE SEKRET)

$$f(x) = \sum 8 \cdot \left(\frac{x-2}{1-2}\right)\left(\frac{x-4}{1-4}\right) + 6\left(\frac{x-1}{2-1}\right)\left(\frac{x-4}{2-4}\right) +$$

$$+ 10\left(\frac{x-2}{4-2}\right)\left(\frac{x-4}{4-4}\right) =$$

$$= \cancel{8(-x+2)(x-4) + 6(x-1-2(x-4))} \quad \frac{8(x-2)(x-4)}{3} - 3\frac{(x-1)(x-4)}{1} -$$

$$- 5\frac{(x-1)(x-2)}{3} = \frac{8}{3}(x^2-4x-2x+8) - 3(x^2-4x-x+4) + \frac{5}{3}(x^2-3x+2) =$$

$$= 7(x^2-6x+8) - 3(x^2-5x+4) + 6(x^2-3x+2) =$$

$$= 7x^2 - \frac{7 \cdot 6}{\mathrm{m}(13)}x + 4 - 3x^2 + 2x + 1 + 6x^2 - 5x + 12 =$$

$\downarrow 3$

$$= 10x^2 + \left(\frac{-6}{\mathrm{mod}\,13}\right)x + 4$$

$\Downarrow 7$

do wyznaczenia

Funkcja jednokierunkowa

$F: X \to Y$

1) obliczanie $F(x)$ "łatwe"

2) obliczanie $F^{-1}(x)$ "trudne"

Dane: p, q - liczby pierwsze

WYNIK: $n = pq$

$F(x,y) = xy$

$F(p,q) = n, \quad n = pq$

$\Downarrow$

$O(\log^2 p)$

1) $f(n) = O(g(n)) \Leftrightarrow \exists_{c>0} \; |f(n)| \leq c g(n)$

* $f(n)$ - liczba bitów n

$f(n) = O(\log n)$

* $f(n) =$

Algorytm działający na k bitach:

jest nazwa niedeterministycznego (wykładniczego)

jeśli liczba operacji elementarnych na bitach potrzebnych do wykonania tego algorytmu jest rzędu $O(k^c)$

w przyp. $\left[\left(e^{ck}\right)\right] \underset{\text{stało}}{\underset{\Uparrow}{c>0}}$

OP. ELEMENT. NA BITACH

0000	1111
0011	0011
0101	0101
0110	1001
0001	0114

np.

```
  1 1 1
  1 1 0 1
  1 0 1 1
-------------
  1 1 0 0
```

→ liczba operacji $a \cdot b = O(\log b)$, $b \geq a$

$q^2 \leq pq \leq p^2$

$q \leq \sqrt{n}$

2 | n

3 | n

$[\sqrt{n}]$ | n

```
  1 1 0 1
  1 0 1 1
-------------
  1 1 0 1
  1 9 0 1
  0 0 0 0
1 9 0 1 4
```

liczba operacji $a * b = O(\log^2 b)$

$O(\sqrt{n}\log^2 n) \cdot O(\sqrt{n}\log n)^2 =$

$= O(n^{\frac{1}{4}} \log n)^2 = O\left(e^{\log(n^{\frac{1}{4}}\log n)^2}\right)$

liczba

BPSI - 24/10

p, q p ≠ q - l. pierwsza

(1) $f(p,q) = pq = n$ $O(\log^2 p)$

p≠q (2) $f^{-1}(n) = r$, $r | n$ $O(e^{\frac{1}{2}\log n})$

Przypuszcza się, że ten problem jest trudny ale nieudowodniony.

→ Kandydat na f. jednokierunkową

Niech p będzie liczbą pierwszą

$\mathbb{Z}_p^* = \{1, 2, \dots, p-1\}$

Element $g \in \mathbb{Z}_p^*$; $\mathbb{Z}_p = \{g^i\} \bmod p$, $i = 1, 2 \dots p-1\}$

nazywamy generatorem $\mathbb{Z}_p^*$

Niech p będzie liczbą pierwszą, $\mathbb{Z}_p^*$ g-generator $x \in \langle 1, p)$

1) $f(p, g, x) = g^x \bmod p = y$

2) $f^{-1}(p, g, y) = x$ taki, że $g^x \equiv y \bmod p$

$g^x = \underbrace{g \cdot \dots \cdot g}_{x} \bmod p$ ZŁE PODEJŚCIE

$g^{101_2} = g^5 = g^4 \cdot g = (g^2)^2 g$ → jedno podnoszenie do kwadratu, przemnożenie przez g

g^{11} = ... = ... met. (g^n) jedno.

jedno do kwadr. · g ; 5 mnożeń ; 3 mnożenia ; 10 mnożeń ; 5 mnożeń

$= ((g \bmod p)^2 \bmod p) \, g \bmod p$

Idea algorytmu potęgowania metody iteracyjnego podwajania do kwadratu

$g^{101} \bmod p$

$g^1 \equiv g \bmod p$

$g^{(10)_2} \equiv g^2 \bmod p$

1) $g^{(100)_2} \equiv (g^{10_2})^2 \equiv (g^2)^2 \bmod p$ $\quad O(\log^3 p)$

2) $g^{101_2} = (g^{(100)_2})g \equiv (g^2)^2 \cdot g \bmod p$

$O(\log p)$ + liczba bitów $\cdot$ $O(\log^2 p)$ ↓ koszt mnożenia $= O(\log^3 p)$

Funkcja 1) $\Leftrightarrow$ koszt alg. $O(\log^3 p)$

Funkcja 2) $(-)$ koszt alg.

$$\mathbb{Z}_p^* = \{ g^i \bmod p \quad i = 1, 2, \dots, p-1 \}$$

$$O(p \log^3 p) = O((p^{\frac{1}{3}} \log p)^3) = O(e^{\frac{1}{3} \log p})$$

ASYMETRYCZNY PROTOKÓŁ SZYFROWANIA (Pr. z kluczem pub.)

1. Generowanie kluczy
2. Alice wybiera asym. met. szyf. (E, D)
3. Generuje dwa klucze do tej met.:

4) klucz K_A - publiczny
a klucz k_a - tajny
$K_A \Rightarrow$ klucz do szyfrowania - klucz publiczny
$k_a \Rightarrow$ - " - tajny

Zał: Zakładamy, że K_A nie jest możliwe w praktyce obliczenie k_A

(E, K_A) — NIE MA PEWNOŚCI ŻE TEN KLUCZ NALEŻY DO ALICE

ALICE

k_A

1) Deszyfruje C ten
$M = D_{k_A}(c)$

C

BOB

1) Chce wysłać m do Alice
2) Pobiera klucz publiczny
3) Szyfruje M ten
$C = E_{K_A}(M)$
4) Wysyła M do Alice

Wady protokołu:

schem. 2) Nie ma pewności, że ten K_A jest Alice
schem. 3) E, D działają WOLNO

Zalety:
* Każdy może napisać do Alice i nie trzeba przechowywać wielu kluczy tajnych

$E, D, K_A \xrightarrow{!} k_a$
$C_1, \ldots C_n$

Problem logarytmu Dyskretnego w Z_p^*

Dane: p, g, y $\quad g, y \in Z_p$ $\quad g$ - generator

WYNIK: x: $\quad g^x \equiv y \pmod p$ Jeśli jest brak g to g musi być dr to czyli mały

$a \in Z_p^*$ $\quad ord_p(a) = r$ $\quad a^r \equiv 1 \pmod p$ $\quad r$ - najmniejsze takie potęga

PROTOKÓŁ DIFFIEGO-HELMANA

Cel: Ustalenie wspólnego sekretu S

OBIE STRONY MAJĄ WPŁYW NA SEKRET

ALICE		BOB
S - sekret		[p, g] → x - tajny
1) Losuje liczbę pierwszą p	PUB	1) losuje x_B $\quad 1 < x_B < p$
$g \div g \in Z_p^*$		2) Oblicza $y_B = g^{x_B} \bmod p$
2) Losuje x_A z przedziału od (1 do p)	PRYW	3) y_B do Alice
3) Wysyła [p i g] do Boba	[p, g]	4) Oblicza $S = y_A^{x_B} \bmod p$
4) Oblicza $y_A = g^{x_A} \bmod p$	y_A	
5) Wysyła y_A do Boba	y_B	
6) Oblicza $S = y_B^{x_A} \bmod p$		

klucz-sekret

S - tajny klucz

$$S = y_B^{x_A} = y_A^{x_B}$$

$$g^{x_B x_A} = g^{x_A x_B}$$

OBOJE MAJĄ TĄ SAMĄ LICZBĘ.

PASYWNY ATAK: EVE: Co zna EVE? p, g, y_A, y_B mimo 3.

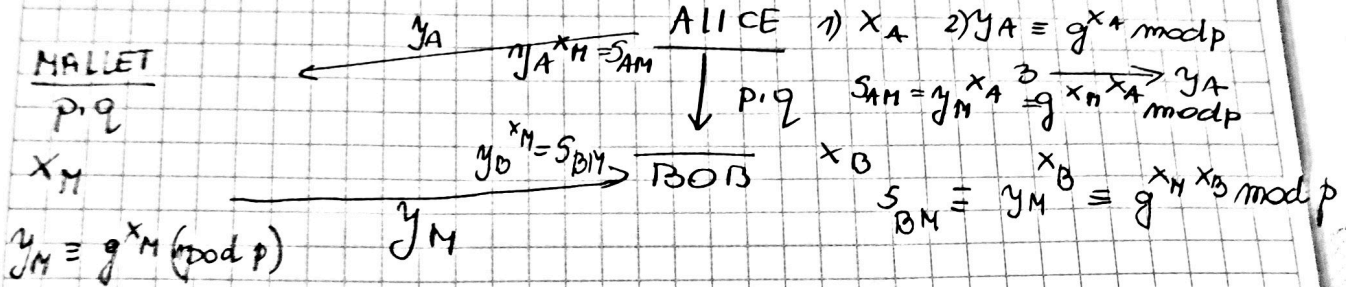
1° mp. z y_A obliczyć $x_A \Rightarrow$ DLP (problem log. dyskret.)

(ZDH) 2° $y_A, y_B \quad g^{x_A}, g^{x_B} \longrightarrow g^{x_A x_B}$

Zał. Diffiego - Hellmana (ZDH) } Pokazano że jak ktoś umie tel DLP to umie tel ZDH

D: y_A, y_B, p, q

W: S



BSP1

PROTOKÓŁ RSA

$p > q$

$\log p \approx 1024$

ALICE

1) losuje p, q, $p \neq q$ (nie za blisko ani nie daleko) liczby pierwsze

2) Oblicza $n = p \cdot q$ (pub, tajne)

$\varphi(n) = (p-1)(q-1)$

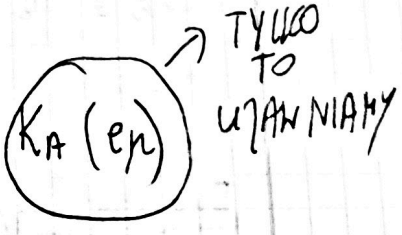
FUNKCJA EULERA

3) Losuje $e < \varphi(n)$: $(e, \varphi(n)) = 1$ (takie, że; pierwsze)

4) Oblicza d takie, że $ed \equiv 1 \pmod{\varphi(n)}$ (tajne)

5) $K_A = (e, n)$ - klucz do szyf. (pub)

$k_A = (d, n)$ - klucz do deszyf. (tajny)



Szyfrowanie

ALICE

1) Deszyfrowanie

$M \equiv C^d \pmod{n}$

E: BOB

$M < n$

1) Oblicza $c \equiv M^e \pmod{n}$

EVE:

$$\& \begin{cases} \varphi(n) = (p-1)(q-1) = pq - p - q + 1 = n - p - q + 1 \\ n = pq \end{cases}$$

$\Rightarrow$ TYM BYŚMY MOGLI OBLICZYĆ P I Q

n to $\varphi(n)$ $\Leftarrow$ TO JEST TRUDNE DO OBLICZENIA

$C^{\frac{1}{e}} \bmod n$ jest trudne

$x^e = c$

$x^2 = 1 \bmod p \quad \begin{pmatrix} 1020 \\ -1,1 \end{pmatrix}$

$x^2 = 1 \bmod n$

$(\pm x_0)^2 \equiv 1 \bmod n$

$x_0^2 \not\equiv -1 \bmod n$

$C_1 \equiv M_1^e \bmod n$

$C_2 \equiv M_2^e \bmod n$

$C_1 C_2 = (M_1 M_2)^e$

ElGamala

Generowanie kluczy

Alice

1) Losuje liczbę pierwszą
$\log p \geq 1024$

2) Losuje $g \in \mathbb{Z}p^*$ - generator $\mathbb{Z}p^*$

3) losuje $x_A : 1 \leq x_A \leq p-1$

4) oblicza $y_A = g^{x_A} \bmod p$

5) $K_A = (p, g, y_A)$ => public key

6) $k_A = (p, g, x_A)$ => klucz tajny

ElGamal

$K_A = (p, g, y_A)$

Szyfrowanie

D: ALICE ($k_A = p, g, x_A$)

1) $M = c_2 c_1^{-x_A} \bmod p$

E: BOB

$0 \le M \le p$

0) Pob. k_A

1) losuje k, $1 < k < p-1$

2) oblicza $c_1 \equiv g^k \bmod p$

3) oblicza $c_2 = M y_A^k \bmod p$

4) $C = [c_1, c_2]$

Poprawność ElGamala

$$c_2 c_1^{-x_A} \equiv M y_A^k (g^k)^{-x_A} \equiv M g^{x_A k} g^{-x k x_A} \equiv M \pmod p$$

EVE: $K_A = (p, g, y_A) \xrightarrow{DLP}$ problem dyskretnego x_A

$y_A, c_1 \xrightarrow{DH} g^{x_A k} \bmod p$

$y_A \equiv g^{x_A}$ $c_1 = g^k$

PODPIS CYFROWY

$sig_k($ H(M) $) = S$ (conajmniej 160 bitów)

$VER_k(H(M), S) \in$ TAK / NIE

K

[M, S]

3 sposoby kiedy podpis tu nie zgodzi (VER)

ALGORYTM PODPISU

$SIG_{k_A}(H(M)) = S$

↑ klucz tajny do weryfikacji podpisu

[SIG, VER]

klucz pub. do weryfikacji ← $VER_{K_A}(H(M), S) = \begin{cases} TAK \Leftrightarrow \text{akcept. podp.} \\ NIE \Leftrightarrow \text{nie H. podpisu} \end{cases}$

PROTOKÓŁ PODPISU CYFROWEGO

ALICE

1) ALICE wybiera [SIG, VER), H

2) Generuje klucz pubby do
   Sig
   K_A – klucz p. do weryfikacji
   k_A – klucz tajny do podpisu

3) Publikuje K_A

2

GRUPA ZL*

ALICE PODPISYWANIE WIADOMOŚCI

M - wiadomość

1) Oblina $y = H(M)$
2) oblina $S = sig_{k_A}(y)$
3) Twory $[M, S]$ - podpis pod M

↑ M nie musi być jawne!

WERYFIKACJA POD. BOB

$[M, S]$

1) Pobiera klucz publiczny (osoby)
2) Oblicza $y = H(M)$
3) Jeśli oblicza $W = VER_{K_A}(y, s)$
4) Jeśli W = TAK AKCEPTACJA PODPISU!

PODPIS RSA

$k_A = [e, n]$

SIG:

$S \equiv H(M)^d \bmod n$

$[M, S]$

VER:

$K_A = [e, n]$

1) $y = H(M)$
2) $y' \equiv S^e \bmod n$
3) Jeśli $y = y'$ to W = TAK

klucz tajny
otwarty

PODPIS DSA H = SHA-1

Generowanie klucz DSA

1) Losujemy p i q
liczby pierwsze takie, że
$q | p-1$ i $\log p \geq 1024$
$\log q \geq 160$

ZAIMPLEMENTOWANE ← 2) Znajdujemy $g \in Z_p^*$ rzędu $ord_p(g) = q$

3) LOSUJE X, $1 < x_A < q-1$

4) Oblicza $y_A \equiv G^{x_A} \pmod p$

5) $k_A = [p, q, g, y_A] \Rightarrow$ KLUCZ PUBLICZNY.

$k_A = [p, q, g, x_A] \Rightarrow$ KLUCZ TAJNY do podpisu

PODPISYWANE (DSA) M

SIG: $k_A = [p, q, g, x_A]$

1) M jest dane
Losujemy k, $1 < k < q$

2) Oblicz $r \equiv (g^k \bmod p) \bmod q$

3) Oblicz $s = k^{-1}(H(M) + x_A r) \bmod q$

4) Twory $[M, (r,s)]$ ← podpis pod M

PODOBNIE JAK ELGEL.

BSPI 14/11

DSA (Generowanie klucza)

1) Losujemy p, q takie że $q \mid p-1$
2) Znajdujemy $g \in Z_p$, $ord_p(g) = q$
3) Losujemy x, $1 < x < q-1$
4) Obliczamy $y \equiv g^x \pmod p$
5) $K_A = (p, q, g, y)$ - klucz publiczny
   $k_A = (p, q, g, x)$ - tajny

? + K_{CA} → info l. pub ALICE

DSA (Podpis) M, H - ALICE

$g^k = r \pmod p$

$s k = (H(M) + xr) \mid : s$

$k = s^{-1}(H(M) + xr)$

1) los. k, $1 < k < q$
2) Obl $r \equiv (g^k \pmod p) \bmod q$
3) Obl $s \equiv k^{-1}(H(M) + xr) \bmod q$
4) Para [M, (r, s)] jest podpisem pod M

DSA (veryfication)

1) Jeśli $r > q \vee s > q$ to Bob nie akceptuje podpisu
2) Oblicza $u_1 \equiv s^{-1} H(M) \bmod p$
   $u_2 \equiv r s^{-1} \bmod q$
3) Oblicza $v \equiv (g^{u_1} y^{u_2} \bmod p) \bmod q$
4) Jeśli $v \equiv r$ to Bob akceptuje podpis

POPRAWNOŚĆ ALGORYTMU

$$V \equiv g^{u_1} y^{u_2} \equiv g^{u_1} g^{x u_2} = g^{u_1 + x u_2} \equiv g^{s^{-1} H(M) + x r s^{-1}} \equiv$$

$$\equiv g^{s^{-1}(H(M) + xr)} = g^k \equiv r \pmod p \quad \square$$

SHOT ON MI 8
AI DUAL CAMERA

①

prosi o odlewnicie zi i7i ③

PS

CERTYFIKAT, to podpis pod X.

X - klucz publiczny do alg. asymetrycznego
- -||- do weryfikacji podpisu
- -||- do uprawnień
- parametry DH = [p, q, y]
- inne

CERTYFIKATY X509 [SIG, VER]

unieważnione certyfikaty wrzuca

CRL

REPOZYTORIUM

CERT A
CERT B

k_{CA} CA - ZAUFANY SERWER

k_{CA} - klucz tajny do podpisu

CERT = [X, S] $S = sig_{k}(x)$

PROTOKÓŁ WYSTAWIANIA CERTYFIKATU KLUCZA PUB.

ALICE

1) Alice generuje ~~syst~~ alg. z kluczem publicznym [E, D]

2) Generuje klucze ob [E, D] (k_A, k_A)

3) Tworzy request

$REQ_A = [E_{ID}, K_A, A_{ID}, K_A]$

pewne id (inform.)

4) REQ do CA

5) CA sprawdza REQ_A

6) Tworzy i podpisuje cert request osoba

$S = SIG_{k_{CA}}(REQ_A)$

7) Tworzy certyfikat

$CERT_A = [E_{ID}, A_{ID}, K_A, S]$

8) Umieszcza certyfikat w REP.

9) Wysyła CERT do Alice.

10) "Bezpieczeństwo" przekazuje k_{CA} Alice.

BSPI – 21. 11. 2019

PROTOKÓŁ SYGNOWANIA Z KLUCZEM PUB Z CERTYFIKATAMI

ALICE

(k_{CA}) BOB (1)

1) Pobiera cert. Alice z repo
2) Weryfikuje $CERT_{CA}$ kluczem k_{CA} — jeśli weryfik. jest pop. „ok" to ufa kl. pub Alice
3) to ufa k_A z CERT
4) Wykonuje protokół z kluczem publicznym

(CA_1) k_{CA_1}

REPO
$CERT_{CA1}$
$CERT_{CA2}$
$CERT_A$
$CERT_B$

(CA_2) k_{CA_2}

BOB k_{CA_2}

1) Wer. $CERT_{CA_1}$
2) Ufa k_{CA1}
3) ~~Ufa~~ Wer. $CERT_A$
4) Ufa k_A

5.12 => KOLOKWIUM POŁÓWKOWE !

ŚLEPE PODPISY

ALICE

$M_1\ R_1$ ☐
$M_2\ R_2$ ☐
⋮ ⋮ ⋮
$M_k\ R_k$ ☐

np. imiona – numery – ZACIEMNIONE PEŁNE OBSZARY

(M)

NOTARIUSZ

BOB

(Z_i) → zawiera z przyjm. prawd. takie same informacje

$Z_1, \dots Z_k$

1) losuje i oraz odkłada Z_i
2) wysyła „i" do Alice i prosi o odkrycie z_j $i \neq j$ (3)

~~wysyła~~ odkrywa

WŁASNOŚĆ FUNKCJI FIG

f - funkcja zaciemnienia

g - funkcja, która odciemnia

$$g(SIG_K[f(M)]) = SIG_K(M)$$

BSPI –

f - f.

g - f

M - h

g

Algo

K =

k =

f:

1)

$\forall$

$x \in \mathbb{Z}_r$

g

t

xe

p

1

2

3

BSPI - wykład 21.11.2019

f - f szyfrująca

g - f odszyfrująca

M - wiadomość

$$g(SIG_k(f(M))) = SIG_k(f(M))$$

Algorytm Chauma

$K = (n, e)$ - klucz do weryfikacji podpisu RSA

$k = (d, n)$ - klucz do podpisu RSA

$f: Z_n \to Z_n$ (f. szyfrująca)

1) Losuje $k \in Z_n$ (tajne), $k \neq 0$, $(k, n) = 1$ ⇒ muszą być względnie pierwsze

$\forall_{x \in Z_n} \; f(x) = x k^e \pmod n$

$g: Z_n \to Z_n$ dla $k \in Z_n$, $k \neq 0$

$\forall_{x \in Z_n} \; g(x) = k^{-1} x \bmod n$

PROTOKÓŁ CHAUMA

ALICE

CHCE ŚLEPO PODPISAĆ M

1) Pobiera $k_B = (n, e)$
2) Losuje $0 < k < n$ $(k, n) = 1$
3) Oblicza $y \equiv M k^e \pmod n$ (Zaślepia M) np. $y \equiv H(M) k^e \pmod n$ (też musimy wysłać M a Bob musi obliczyć hash.)
4) Wysyła y do BOBA
7) odkrywa podpis pod M, tzn. oblicza $s \equiv k^{-1} \pmod n$

$k_B \in (d, n)$ - do podpisu

$K_B = (n, e)$ - do weryf

$k_B =$

BOB

(notariusz - os. która świad. podpisy)

5) Podpisuje ślepo y tzn. oblicza (Podpis RSA)
$z \equiv y^d \pmod n$ (1)
6) Wysyła z do Alice
8) Tworzy [M, s]

1) Jeśli $b_i = 0$ to Alice (3)
wysyła mu L_j

poprawność podpisu ślep.

$$k^{-1} z \equiv k^{-1} y^d \equiv k^{-1}(M^* k^e)^d = k^{-1} M^d k^{ed} \equiv k^{-1} \cdot M^d \cdot k = M^d \equiv$$
$$\equiv s \pmod n$$

PROTOKÓŁ REALIZUJĄCY CYFROWY BANKNOT

FAZY PROTOKOŁU

I Wydanie banknotu B przez Bank.

II Zakup Alice za pomocą BANKNOTU B u Boba (sprzedawca)

III Złożenie banknotu B w Banku przez Boba.

Protokół zapewnia anonimowy zakup przez Alice, pod warunkiem, że ona nie oszukuje

Wykrywa oszustwo Boba jeśli chciałby złożyć ~~ten sam~~ banknot w banku ponownie.

(I) $B_i =$ | y | x_i | L_{i1} L_{i2} | R_{i1} R_{i2} |

$i = \{1 \dots 100\}$

L_{i100} R_{i100}

kwota Boba — losowa liczba (kolejna cyfra nr. seryjny)

I_A - ciąg, który identyfikuje Alice.

1) wykonuje prot. podpisu ślepego

a) losuje R_{ij} → mamy ustalone (bo to numer banknotu)

b) obl $L_{ij} = I_A$ XOR R_{ij} → $j \in \{1 \dots 100\}$

c) $R_{ij} L_{ij}$

decimy po j po kolei ale do końca na ustalonych „i"

2) Dla ustalonego i-tego banknotu, dla $j = 1 \dots 100$ wybieramy zobowiązania bitowego dla

(R_{ij}) oraz niezależnie L_{ij} I fana algorytmu zobowiązania bitowego.

„inf" o zdarzeniu bitowym
$([R_{ij}])$ - oznaczenie „-"
(y_j, r_j)

3) Dla każdego „i" Alice zalicza B_i $i = \{1 \dots 100\}$ — cały Banknot
tzn. $y_i \equiv f(B_i)$

4) Wysyła y_i do Banku
$\forall i \in \{1 \dots 100\}$

5) Bank losuje $j \in \{1 \dots 100\}$ i prosi Alice o odkrycie Banknotu B_i $i \neq j$ (wszystkie pozostałe) i sprawdza czy wszystkie banknoty czy są prawidłowo zbudowane — y_i, x_i, L_{ij}, R_{ij} oraz obliczą wszystko L_{ij}
w przeciwnym wypadku odmawia wysłania Alice.

6) Bank ślepo podpisuje
y_j tzn. oblicza $z_j \equiv y_j^{d} \pmod n$, gdzie $k_{BANK} = (n, d)$ ← TAJNY

7) Alice odbiera podpis pod B_j z z_j w ten sposób otrzymuje $[B_j, S_j]$ S_j - jest podpisem pod B_j

(II)

ALICE

1) Wysyła (B_j, S_j) do Boba

BOB
(sprzedawca)

2) Weryfikacja podpisu Banknotu (z danymi Banku) Jeśli weryf. jest nie poprawna to sorry kończy

3) Losuje 100 bitów
$(b_1, b_2 \dots b_{100})$

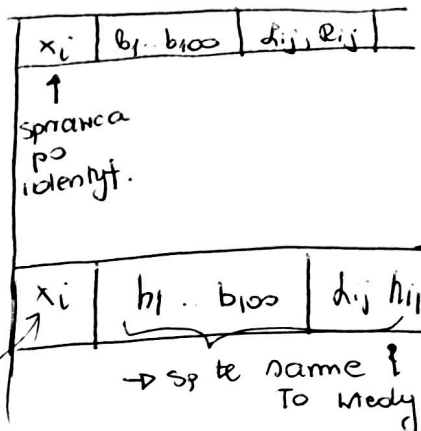
4) Jeśli $B_i = 0$ to Alice wysyła mu L_{ij}
otrzyma

5) Jeśli $b_i = 1$ to odtwarza R_{ij}

BOB

1) Wysyła $[B_j, s_j], (b_1 .. b_{100})$, oraz odpowiadające d_{ij}, R_j

BANK



→ są te same ! To wtedy SPRZEDAWCA OSZUKAŁ

po samym identyf. nie może stwierdzić kto oszukał ale

→ ale jeśli te $(b_{ij}, R_{ij} L_{ij})$ się różnią to wtedy ALICE oszukała