Securitatea Sistemelor Informatice

- Curs 6.1 -
Padding-oracle attack

Adela Georgescu

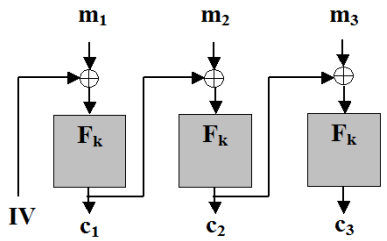
Facultatea de Matematică și Informatică
Universitatea din București
Anul universitar 2022-2023, semestrul I

Atacul bazat pe oracol de padding

- In cursul anterior am discutat despre securitate CCA
- Motivăm importanța securității CCA arătând un atac devastator din viața reală
- Mai mult, atacul cere ca un adversar să poata afla numai dacă un text criptat modificat este unul valid (care se poate decripta corect), nefiind necesară întreaga funcționalitate a unui oracol de decriptare (care întoarce textul clar corespunzător unui text criptat).
- Acest fapt poate fi exploatat pentru aflarea întregului text clar

Atacul bazat pe oracol de padding

- Am văzut cum funcționeaza modul CBC când lungimea mesajului clar este multiplu de lungimea L blocului de criptat (suportat de F_k) in octeți



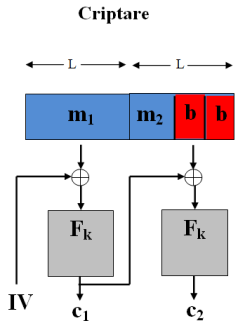
Decriptare

$$m_1 = F_k^{-1}(c_1) \oplus IV$$
$$m_2 = F_k^{-1}(c_2) \oplus c_1$$
$$m_3 = F_k^{-1}(c_3) \oplus c_2$$

- Ce se întâmplă când $|m| \neq L$?
- Folosim padding-ul *PKCS#7* :
  - Fie b numărul de octeți de adăugat la ultimul bloc pentru a avea $|m|$ multiplu de L ($1 \leq b \leq L$)
  - Se adaugă b octeți la ultimul bloc din m, fiecare reprezentând valoarea lui b

CBC cu padding PKCS7

Considerăm situația în care un client trimite mesaje criptate în modul CBC către un server.

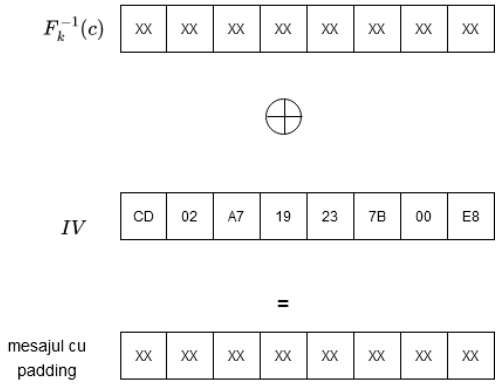


- în urma decriptării se obțin $m_1||m_2$
- se citește octetul final b
- dacă ultimii b octeți au toți valoarea b, atunci se scoate padding-ul și se obține mesajul original m
- altfel întoarce mesajul *padding gresit* și cere retransmiterea mesajului

Server-ul acționeaza ca un oracol de padding - adversarul îi trimite texte criptate și află dacă padding-ul este corect sau nu

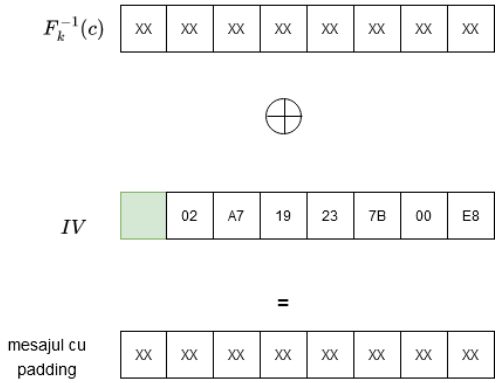
Ideea atacului cu oracol de padding

- Unui text criptat (IV, c) îi corespunde textul clar cu padding $m' = F_k^{-1}(c) \oplus IV$
- Dacă un adversar modifică octetul i din IV atunci modificarea se va reflecta și în octetul i din m'



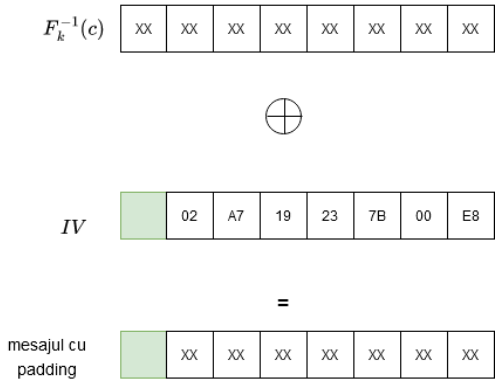
Ideea atacului cu oracol de padding

Adversarul modifica primul octet din IV



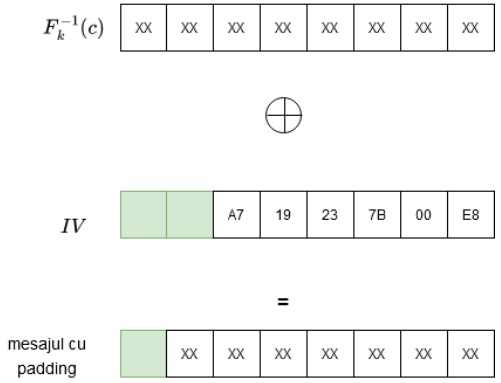
Ideea atacului cu oracol de padding

Modificarea se reflectă în primul octet din mesajul cu padding; apoi trimite mesajul (IV', c) și verifică dacă primește eroare



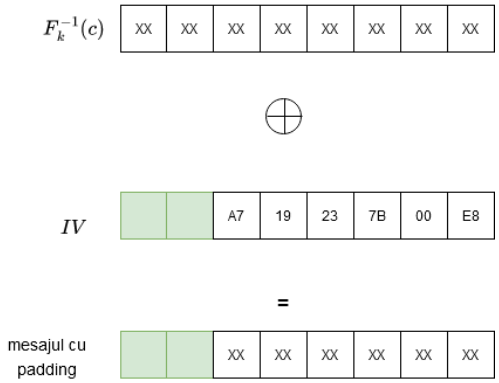
Ideea atacului cu oracol de padding

In caz contrar, adversarul modifică al 2-lea octet din IV



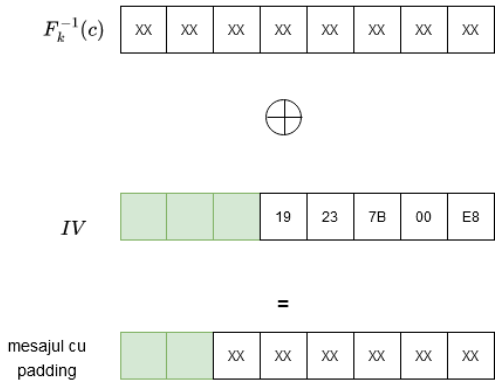
Ideea atacului cu oracol de padding

Modificarea se reflectă în al 2-lea octet din mesajul cu padding; apoi trimite mesajul (IV', c) și verifică dacă primește eroare

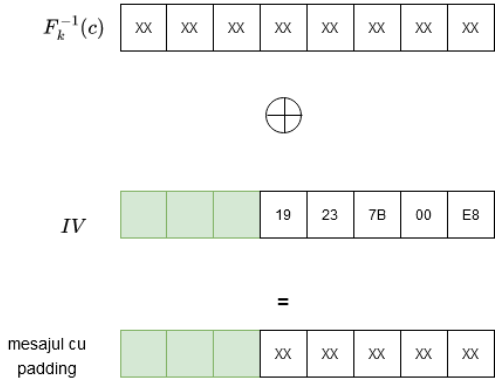


Ideea atacului cu oracol de padding

In caz contrar, adversarul continua cu al 3-lea octet din IV

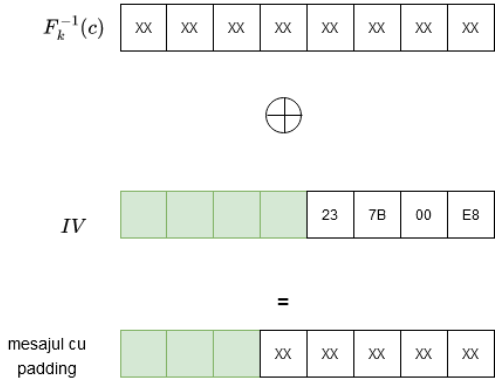


Ideea atacului cu oracol de padding

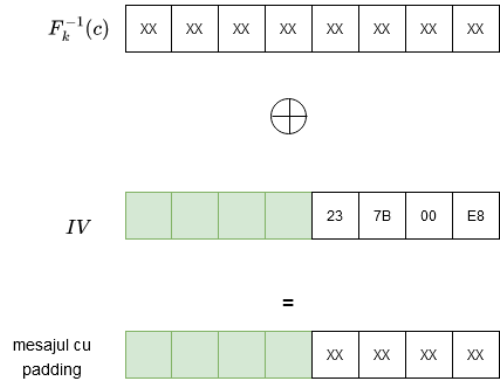


Ideea atacului cu oracol de padding

In caz contrar, adversarul continuă și cu al 4-lea octet din IV

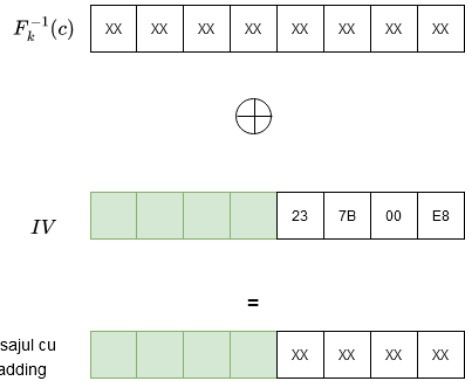


Ideea atacului cu oracol de padding



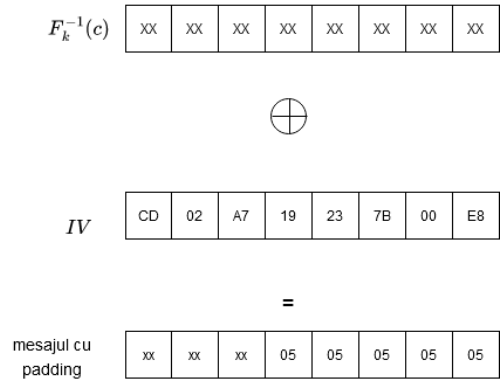
- Eroare la decriptare

Ideea atacului cu oracol de padding

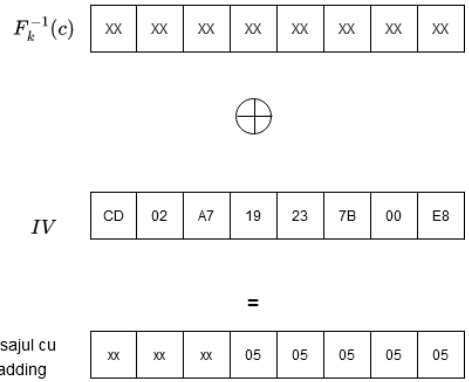


- Eroare la decriptare
- adversarul deduce ca oracolul verifica ultimii 5 octeti, care au valoarea 05

Ideea atacului cu oracol de padding



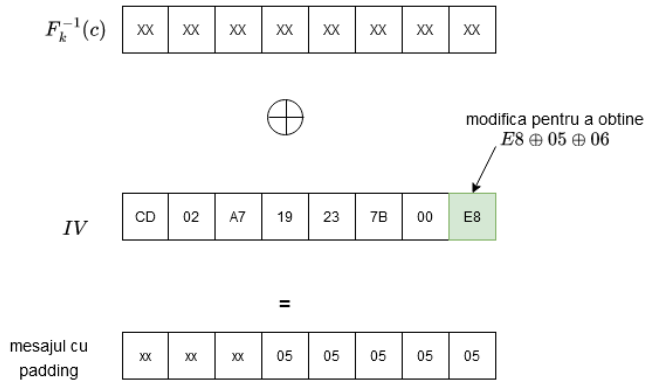
Ideea atacului cu oracol de padding



Primii 3 octeti din mesajul cu padding sunt încă necunoscuți atacatorului

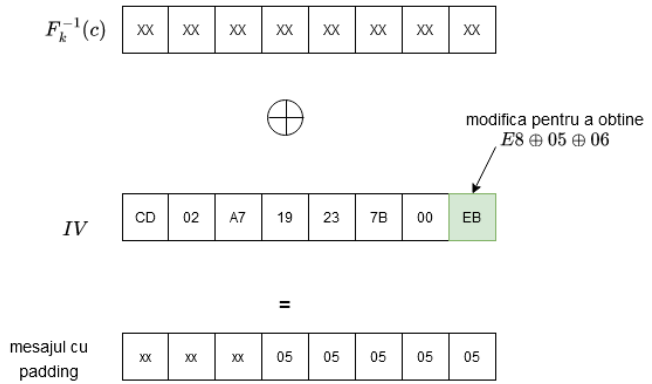
Ideea atacului cu oracol de padding

Adversarul încearcă să găsească primii 3 octeți din mesajul cu padding



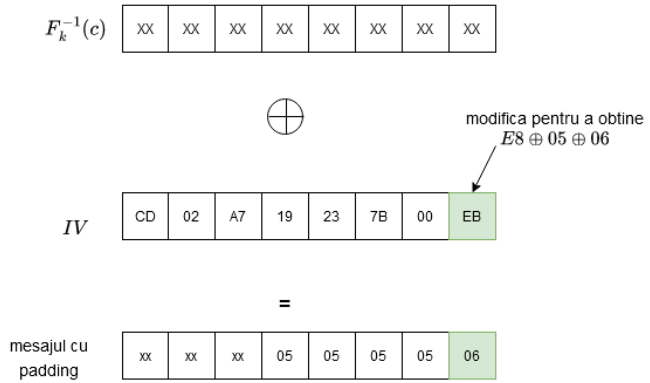
Ideea atacului cu oracol de padding

Adversarul încearcă să găsească primii 3 octeți din mesajul cu padding
El modifica cel mai din dreapta octet din IV



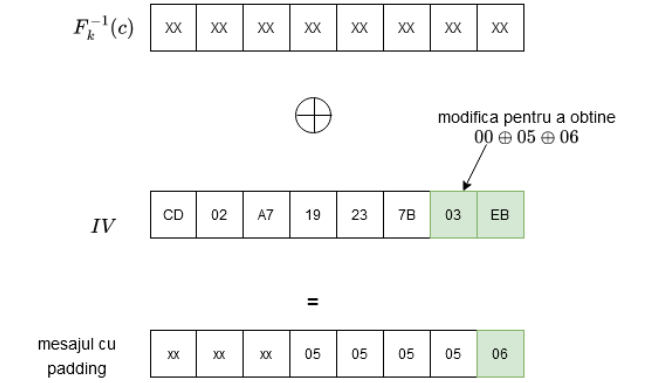
Ideea atacului cu oracol de padding

Adversarul încearcă să găsească primii 3 octeți din mesajul cu padding
Modificarea se va reflecta in cel mai din dreapta octet din mesajul cu padding

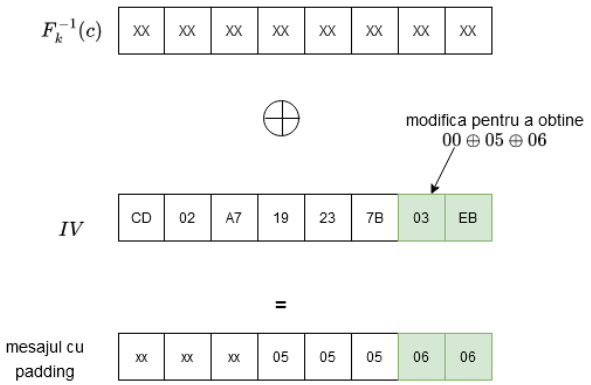


Ideea atacului cu oracol de padding

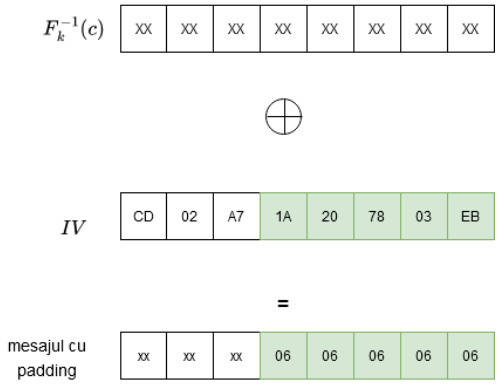
Adversarul va proceda similar pentru ceilalti octeti



Ideea atacului cu oracol de padding

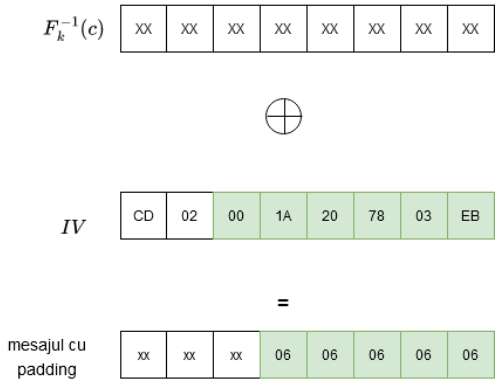


Ideea atacului cu oracol de padding



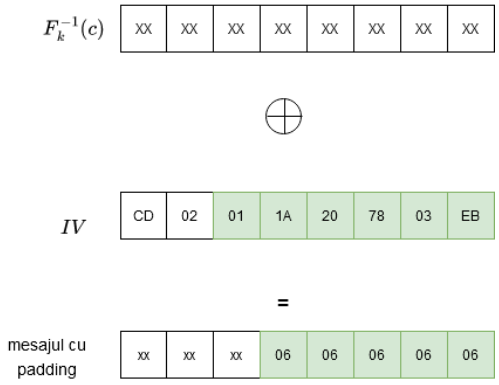
Dacă adversarul trimite acest IV împreună cu c, sunt șanse mici să nu primească eroare la decriptare

Ideea atacului cu oracol de padding



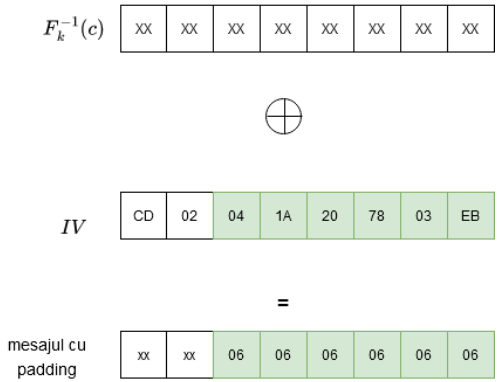
Va incerca pe rand toate valorile posibile pentru al 3-lea octet din IV ....

Ideea atacului cu oracol de padding



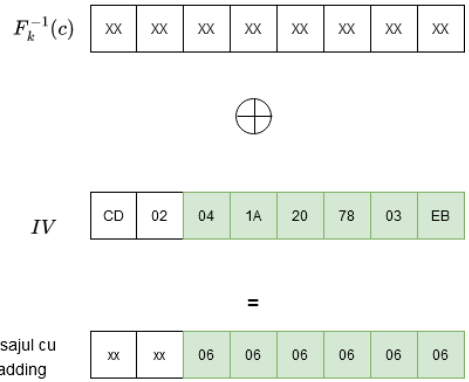
Va încerca pe rând toate valorile posibile pentru al 3-lea octet din IV ....

Ideea atacului cu oracol de padding



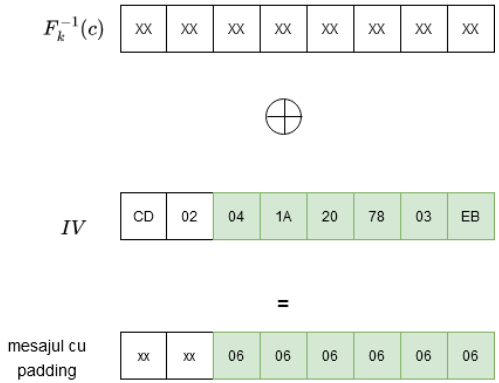
Până cand decriptarea va funcționa; când aceasta se întâmplă, al 3-lea octet din mesajul cu padding este 06 (doar atunci decriptarea funcționează)

Ideea atacului cu oracol de padding



Acum A cunoaște $xx \oplus 04 = 06$ și deci el poate calcula $xx \oplus A7$ (valoarea inițială a mesajului cu padding pe octetul 3).

Ideea atacului cu oracol de padding



Adversarul poate repeta același proces pentru a afla al doilea octet și apoi primul din mesajul cu padding.

Complexitatea atacului cu oracol de padding

- sunt necesare cel mult L incercări pentru a afla b
- cel mult $2^8 = 256$ încercări pentru a afla fiecare octet din mesajul inițial
- în total sunt necesare 256 * *bt* încercări (unde *bt* reprezintă numărul de octeți din mesajul original) pentru a găsi întregul text clar

Securitatea Sistemelor Informatice

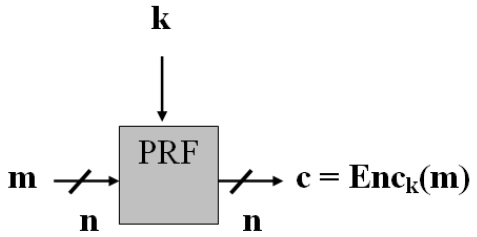
- Curs 6.2 -
Construcții practice PRF

Adela Georgescu

Facultatea de Matematică și Informatică
Universitatea din București
Anul universitar 2022-2023, semestrul I

Sisteme bloc ca PRF

- Am văzut că sistemele de criptare bloc folosesc *PRF*;



Sisteme bloc ca PRF

- În criteriile de evaluare pentru adoptarea AES s-a menționat:

  The security provovided by an algorithm is the most important factor... Algorithms will be judged on the following factors...

  The extent to which the **algorithm output is indistinguishable from a random permutation on the input block**.

- Întrebare: Cum se obțin *PRF* în practică?

Paradigma confuzie-difuzie

- Se construiește funcția F, pe baza mai multor funcții aleatoare f_i de dimensiune mai mică;
- Considerăm F pe 128 biți și 16 funcții aleatoare $f_1, \ldots, f_{16}$ pe câte 8 biți;
- Pentru $x = x_1 || \ldots || x_{16}$, $x \in \{0,1\}^{128}$ $x_i \in \{0,1\}^8$:

$$F_k(x) = f_1(x_1) || \ldots || f_{16}(x_{16})$$

- Spunem că $\{f_i\}$ introduc confuzie în F.

Rețele de substituție - permutare

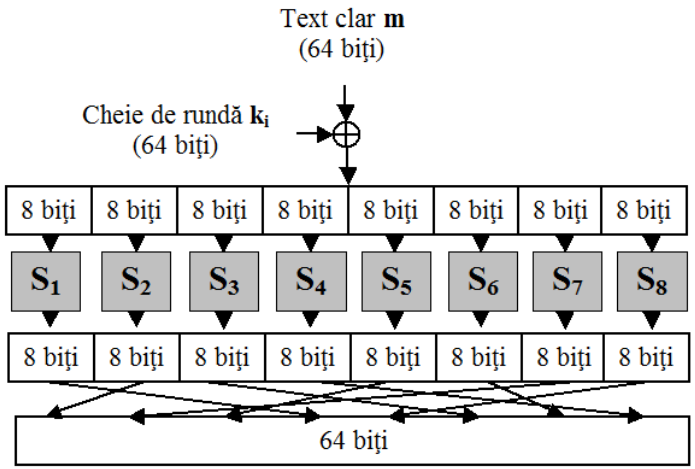
F încă nu este PRF dar

- F se transformă în PRF în 2 pași:
  - **Pasul 1**: se introduce difuzie prin amestecarea (permutarea) biților de ieșire;
  - **Pasul 2**: se repetă o rundă (care presupune *confuzie* și *difuzie*) de mai multe ori;
- Repetarea *confuziei* și *difuziei* face ca modificarea unui singur bit de intrare să fie propagată asupra tuturor biților de ieșire;

Rețele de substituție - permutare

- O rețea de substituție-permutare este o implementare a construcției anterioare de *confuzie-difuzie* în care funcțiile $\{f_i\}$ sunt **fixe** (i.e. nu depind de cheie) si se numesc permutări;
- $\{f_i\}$ se numesc S-boxes (Substitution-boxes);
- Cum nu mai depind de cheie, aceasta este utilizată în alt scop;
- Din cheie se obțin mai multe **chei de rundă** (*sub-chei*) în urma unui proces de derivare a cheilor (*key schedule*);
- Fiecare cheie de rundă este XOR-ată cu valorile intermediare din fiecare rundă.

Rețele de substituție - permutare



Rețele de substituție - permutare

- Există 2 principii de bază în proiectarea rețelelor de substituție - permutare:
  - **Principiul 1**: Inversabilitatea S-box-urilor;
    - dacă toate S-box-urile sunt inversabile, atunci rețeaua este inversabilă;
    - necesitate funcțională (pentru decriptare)
  - **Principiul 2**: Efectul de avalanșă
    - Un singur bit modificat la intrare **trebuie** să afecteze toți biții din secvența de ieșire;
    - necesitate de securitate.

Exemplu: AES - Advanced Encryption Standard

- ianuarie 1997 - NIST anunță competiția pentru selecția unui nou sistem de criptare bloc care să înlocuiască DES;
- septembrie 1997 - 15 propuneri: CAST-256, CRYPTON, DEAL, DFC, E2, FROG, HPC, LOKI97, MAGENTA, MARS, RC6, Rijndael, SAFER+, Serpent, and Twofish;
- 1998, 1999 - au loc 2 workshop-uri în urma carora ramân 5 finaliști: MARS, RC6, Rijndael, Serpent, Twofish;
- octombrie 2000 - după un al treilea workshop se anunță câștigătorul: Rijndael.
- AES este folosit in multe standarde comerciale: IPsec, TLS, IEEE 802.11i (WPA2), SSH, Skype, etc.

AES - Advanced Encryption Standard

[Google Scholar - User profiles]

[http://keccak.noekeon.org/team.html]

Rijndael = **Rij**men + **Dae**men

Descriere AES

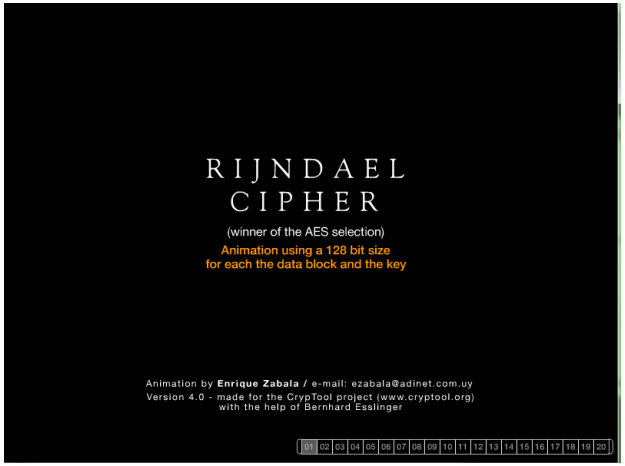
- AES este o rețea de substituție - permutare pe 128 biți care poate folosi chei de 128, 192 sau 256 biți;
- Lungimea cheii determină numărul de runde:

Lungime cheie (biți)	128	192	256
Număr runde	10	12	14

- Folosește o matrice de octeți 4×4 numită **stare**;
- Starea inițială este mesajul clar ($4 \times 4 \times 8 = 128$);
- Starea este modificată pe parcursul rundelor prin 4 tipuri de operații: *AddRoundKey*, *SubBytes*, *ShiftRows*, *MixColumns*;
- Ieșirea din ultima rundă este textul criptat.

Descriere AES

- Rijndael Animation - CrypTool Project:



[http://www.cryptool.org/en/]

Securitatea sistemului AES

- Singurele atacuri netriviale sunt asupra AES cu număr redus de runde:
  - AES-128 cu 6 runde: necesită 2^{72} criptări;
  - AES-192 cu 8 runde: necesită 2^{188} criptări;
  - AES-256 cu 8 runde: necesită 2^{204} criptări.
- Nu există un atac mai eficient decât căutarea exhaustivă pentru AES cu număr complet de runde.

"It is free, standardized, efficient, and highly secure."
(J.Katz, Y.Lindell, *Introduction to Modern Cryptography*)

Rețele Feistel

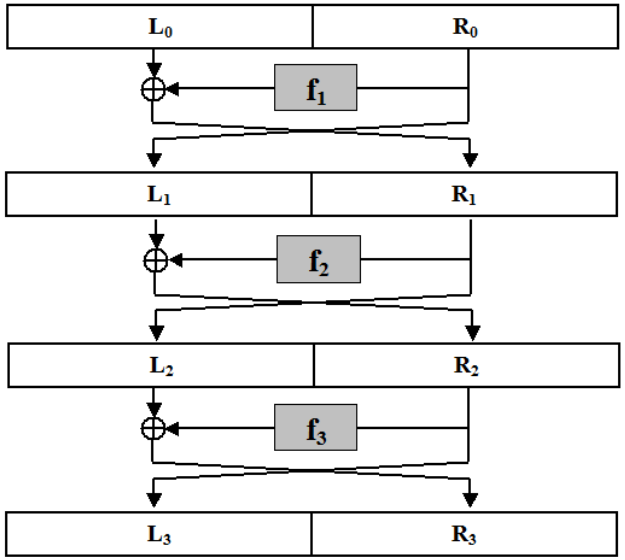
- Se aseamănă rețelelor de substituție-permutare în sensul că păstrează aceleași elementele componente: S-box, permutare, procesul de derivare a cheii, runde;
- Se diferențiază de rețelele de substituție-permutare prin proiectarea de nivel înalt;
- Introduc avantajul major că S-box-urile NU trebuie să fie inversabile;
- Permit așadar obținerea unei structuri *inversabile* folosind elemente *neinversabile*.

Horst Feistel (1915 - 1990)

[Wikipedia]

- Structurile simetrice utilizate în construcția sistemelor bloc poartă numele lui Feistel;
- Munca sa de cercetare la IBM a condus la sistemul de criptare Lucifer și mai târziu la DES.

Rețele Feistel



Rețele Feistel

- Intrarea în runda i se împarte în 2 jumătăți: L_{i-1} și R_{i-1} (i.e. *Left* și *Right*);
- Ieșirile din runda i sunt:

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

$$R_i = L_{i-1} \oplus f_i(R_{i-1})$$

- Funcțiile f_i depind de cheia de rundă, derivând dintr-o funcție publică $\widehat{f_i}$:

$$f_i(R) = \widehat{f_i}(k_i, R)$$

Rețele Feistel

- Rețelele Feistel sunt inversabile indiferent dacă funcțiile f_i sunt inversabile sau nu;
- Fie (L_i, R_i) ieșirile din runda i;
- Intrările (L_{i-1}, R_{i-1}) în runda i sunt:

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

$$L_{i-1} = R_i \oplus f_i(R_{i-1})$$

Securitatea Sistemelor Informatice

- Curs 6.3 -
Data Encryption Standard - DES

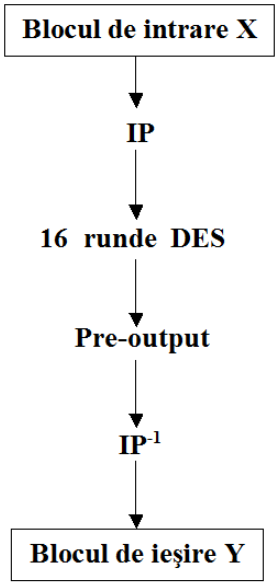
Adela Georgescu

Facultatea de Matematică și Informatică
Universitatea din București
Anul universitar 2022-2023, semestrul I

DES - Data Encryption Standard

- 1970 - Horst Feistel proiectează Lucifer, o familie de cifruri bloc, la IBM, cu lungime cheie = 128 biți și lungime bloc = 128 biți;
- 1973 - NIST inițiază o cerere pentru propuneri în vederea standardizarii cifrurilor bloc în SUA;
  IBM trimite o variantă de Lucifer.
- 1976 - NIST adoptă Lucifer modificat ca standard DES cu lungime cheie = 56 biți și lungime bloc = 64 biți.
- 1997 - DES este spart prin cautare exhaustivă (forță brută).
- 2001 - NIST adoptă Rinjdael ca noul standard AES în locul lui DES.

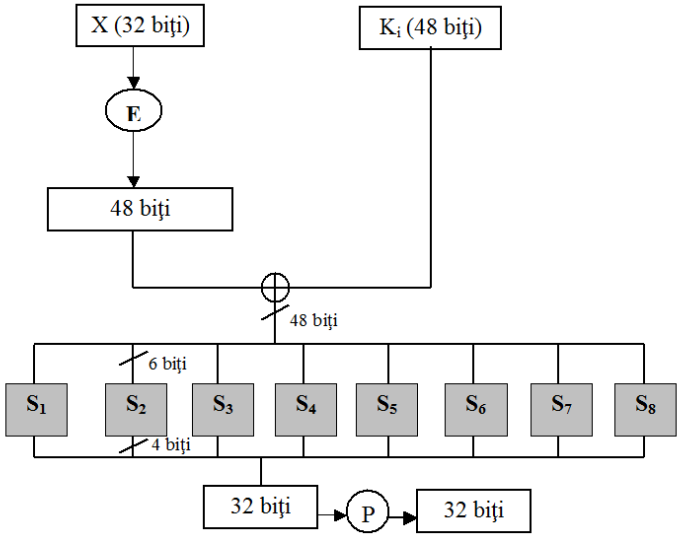
DES



Descriere DES

- DES este o rețea de tip Feistel cu 16 runde și o cheie pe 56 biți;
- Procesul de derivare a cheii (*key schedule*) obține o sub-cheie de rundă k_i pentru fiecare rundă pornind de la cheia master k;
- Funcțiile de rundă $f_i(R) = f(k_i, R)$ sunt derivate din aceeași funcție principală $\widehat{f}$ și nu sunt inversabile;
- Fiecare sub-cheie k_i reprezintă permutarea a 48 biți din cheia master;
- Întreaga procedură de obținere a sub-cheilor de rundă este fixă și cunoscută, singurul secret este cheia master .

Funcția $f(k_i, R)$



Funcția $f(k_i, R)$

- Funcția $\widehat{f}$ este, în esență, o rețea de substituție-permutare cu doar o rundă.
- Pentru calculul $f(k_i, R)$ se procedează astfel:
  1. R este expandat la un bloc R' de 48 biți cu ajutorul unei funcții de expandare $R' = E(R)$.
  2. R' este XOR-at cu k_i iar valoarea rezultată este împărțită în 8 blocuri de câte 6 biți.
  3. Fiecare bloc de 6 biți trece printr-un SBOX diferit rezultând o valoare pe 4 biți.
  4. Se concatenează blocurile rezultate și se aplică o permutare, rezultând în final un bloc de 32 biți.
- **De remarcat:** Toată descrierea lui DES, inclusiv SBOX-urile și permutările sunt publice.

SBOX-urile din DES

- Formează o parte esențială din construcția DES;
- DES devine mult mai vulnerabil la atacuri dacă SBOX-urile sunt modificate ușor sau dacă sunt alese aleator
- Primul și ultimul bit din cei 6 de la intrare sunt folosiți pentru a alege linia din tabel, iar biții 2-5 sunt folosiți pentru coloană; output-ul va consta din cei 4 biți aflați la intersecția liniei și coloanei alese.

S_5		Cei 4 biți din mijloc															
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Biții din margine	00	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

- Modificarea **unui bit** de la intrare întotdeauna afectează cel puțin **doi biți** de la ieșire.
- DES are un puternic efect de avalanșă generat de ultima proprietate menționată mai sus și de permutările folosite;

Securitatea sistemului DES

- Încă de la propunerea sa, DES a fost criticat din două motive:
  1. Spațiul cheilor este prea mic făcând algoritmul vulnerabil la forță brută;
  2. Criteriile de selecție a SBOX-urilor au fost ținute secrete și ar fi putut exista atacuri analitice care explorau proprietățile matematice ale SBOX-urilor, cunoscute numai celor care l-au proiectat.
- Cu toate acestea....
  - După 30 ani de studiu intens, cel mai bun atac practic rămâne doar o căutare exhaustivă pe spațiul cheilor;
  - O căutare printre 2^{56} chei este fezabilă azi (dar netrivială);
  - In 1977, un calculator care să efectueze atacul într-o zi ar fi costat 20.000.000$;

Securitatea sistemului DES

- Primul atac practic a fost demonstrat în 1997 când un număr de provocări DES au fost propuse de RSA Security și rezolvate;
- Prima provocare a fost spartă în 1997 de un proiect care a folosit sute de calculatoare coordonate prin Internet; a durat 96 de zile;
- A doua provocare a fost spartă anul următor în 41 de zile;
- Impresionant a fost timpul pentru a treia provocare: *56 de ore*;
- S-a construit o mașină în acest scop, *Deep Crack* cu un cost de 250.000$

Securitatea sistemului DES

Figure: *Deep Crack-construită pentru căutare exhaustivă DES în 1998*

- Ultima provocare a fost spartă în 22 de ore (efort combinat de la ultimele două provocări);
- Atacurile prin forță brută pe DES au devenit un studiu de caz în încercarea de a micșora costurile;

Securitatea sistemului DES

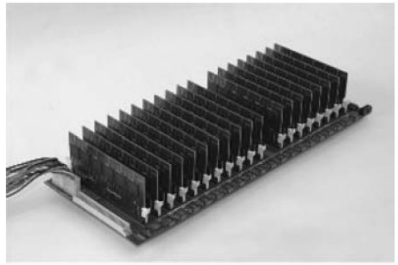


Figure: COPACABANA-Cost Optimized Parallel Code-Breaker

- In 2006, o echipă de cercetători de la universitățile Bochum și Kiel din Germania a construit mașina COPACABANA (*Cost Optimized Parallel Code-Breaker*) care permite găsirea cheii DES cu un timp de căutare mediu de mai puțin de 7 zile, la un cost de 10.000$.

Securitatea sistemului DES

- O altă problemă a sistemului DES, mai puțin importantă, este lungimea blocului relativ scurtă (64 biți);
- Securitatea multor construcții bazate pe cifruri bloc depinde de lungimea blocului;
- În modul de utilizare CTR, dacă un atacator are 2^{27} perechi text clar/text criptat, securitatea este compromisă cu probabilitate mare;
- Concluzionând, putem spune că insecuritatea sistemului DES nu are a face cu structura internă sau construcția in sine (care este remarcabilă), ci se datorează numai lungimii cheii prea mici.

Criptanaliză avansată

- La sfarsitul anilor '80, Biham și Shamir au dezvoltat o tehnică numită criptanaliza diferențială pe care au folosit-o pentru un atac împotriva DES;
- Atacul presupune complexitate timp 2^{37} (memorie neglijabilă) dar cere ca atacatorul să analizeze 2^{36} texte criptate obținute dintr-o mulțime de 2^{47} texte clare alese;
- Din punct de vedere teoretic, atacul a fost o inovație, dar practic e aproape imposibil de realizat;
- La inceputul anilor '90, Matsui a dezvoltat criptanaliza liniară aplicată cu succes pe DES;
- Deși necesită 2^{43} texte criptate, avantajul este că textele clare nu trebuie să fie alese de atacator, ci doar cunoscute de el;
- Problema însă rămâne aceeași: atacul e foarte greu de pus în practică.

Criptanaliza diferențială

- Cataloghează diferențe specifice între texte clare care produc diferențe specifice în textele criptate cu probabilitate mai mare decât ar fi de așteptat pentru permutările aleatoare;
- Fie un cifru bloc cu blocul de lungime n și $\Delta_x, \Delta_y \in \{0,1\}^n$;
- Spunem că diferențiala (Δ_x, Δ_y) apare cu probabilitate p dacă pentru intrări aleatoare x_1, x_2 cu

$$x_1 \oplus x_2 = \Delta_x$$

și o alegere aleatoare a cheii k

$$Pr[F_k(x_1) \oplus F_k(x_2) = \Delta_y] = p$$

- Pentru o funcție aleatoare, probabilitatea de apariție a unei diferențiale nu e mai mare decât 2^{-n};
- La un cifru bloc slab, ea apare cu o probabilitate mult mai mare;

Criptanaliza diferențială

- Dacă o diferențială există cu probabilitate $p \gg 2^{-n}$, cifrul bloc nu mai este permutare pseudoaleatoare;
- Ideea este de a folosi multe diferențiale cu p ușor mai mare decât 2^{-n} pentru a găsi cheia secretă folosind un atac cu text clar ales;
- Criptanaliza diferențială a fost folosită cu succes pentru a ataca cifruri bloc (altele decât DES si AES), de pildă FEAL-8;

Criptanaliza liniară

- Metoda consideră relații liniare între intrările și ieșirile unui cifru bloc;
- Spunem că porțiunile de biți $i_1, \cdots, i_l$ și $i_1', \cdots, i_l'$ au distanța p dacă pentru orice intrare aleatoare x și orice cheie k

$$Pr[x_{i_1} \oplus \cdots \oplus x_{i_l} \oplus y_{i_1'} \oplus \cdots \oplus y_{i_l'} = 0] = p$$

unde $y = F_k(x)$.

- Pentru o funcție aleatoare se așteaptă ca $p = 0.5$;
- Matsui a arătat cum se poate folosi o diferență p mare pentru a sparge complet un cifru bloc;
- Necesită un număr foarte mare de perechi text clar/text criptat.

Creșterea lungimii cheii

- Singura vulnerabilitate practică DES este cheia scurtă;
- S-au propus diverse metode de a construi un sistem bazat pe DES care să folosească o cheie mai lungă;
- Nu se recomandă schimbarea structurii interne întrucât securitatea sistemului ar putea fi afectată;
- Solutie alternativă: considerăm DES o cutie neagră care implementează un cifru bloc "perfect" cu o cheie pe 56 biți.

Criptare dublă

- Fie F un cifru bloc (in particular ne vom referi la DES); definim un alt cifru bloc F' astfel

$$F'_{k_1,k_2}(x) = F_{k_2}(F_{k_1}(x))$$

cu k_1, k_2 chei independente;

- Lungimea totală a cheii este 112 biți, suficient de mare pentru căutare exhaustivă;
- Insă, se poate arăta un atac în timp 2^{56} unde $|k_1| = 56 = |k_2|$ (față de $2^{2 \cdot 56}$ cât necesită o căutare exhaustivă);
- Atacul se numeste meet-in-the-middle;

Atacul meet-in-the-middle

- Iată cum funcționează atacul dacă se cunoaște o pereche text clar/text criptat (x, y) cu $y = F_{k_2}(F_{k_1}(x))$:
  1. Pentru fiecare $k_1 \in \{0,1\}^n$, calculează $z := F_{k_1}(x)$ și păstrează (z, k_1);
  2. Pentru fiecare $k_2 \in \{0,1\}^n$, calculează $z := F_{k_2}^{-1}(y)$ și păstrează (z, k_2);
  3. Verifică dacă există perechi (z, k_1) și (z, k_2) care coincid pe prima componentă;
  4. Atunci valorile k_1, k_2 corespunzătoare satisfac

$$F_{k_1}(x) = F_{k_2}^{-1}(y)$$

  adică $y = F'_{k_1,k_2}(x)$
- Complexitatea timp a atacului este $O(2^n)$.

Criptare triplă

- Există două variante:
  1. Trei chei independente - k_1, k_2 și k_3 iar

$$F'_{k_1,k_2,k_3} = F_{k_3}(F_{k_2}^{-1}(F_{k_1}(x)))$$

  2. Două chei independente - k_1 și k_2 iar

$$F'_{k_1,k_2} = F_{k_1}(F_{k_2}^{-1}(F_{k_1}(x)))$$

- F' este ales astfel pentru a fi compatibil cu F atunci când cheile sunt alese $k_1 = k_2 = k_3$;
- Prima variantă are lungimea cheii $3n$ dar cel mai bun atac necesită timp 2^{2n} (funcționează atacul meet-in-the-middle);
- A doua variantă are lungimea cheii $2n$ si cel mai bun atac necesită timp 2^{2n}.

Triplu-DES (3DES)

- Se bazează pe tripla invocare a lui DES folosind două sau trei chei;
- Este considerat sigur și în 1999 l-a înlocuit pe DES ca standard;
- 3DES este foarte eficient în implementarile hardware (la fel ca și DES) dar totuși lent în implementari software;
- 3DES cu 3 chei încă se mai folosește dar recomandarea este de a înlătura datorită lungimii mici a blocurilor și a faptului ca este lent
- Este popular în aplicațiile financiare și în protejarea informațiilor biometrice din pașapoartele electronice;

Triplu-DES (3DES)

- Singurele dezavantaje ar fi lungimea mică a blocurilor și faptul că este destul de lent fiindcă aplică DES de trei ori;
- Acestea au dus la înlocuirea lui ca standard cu AES;
- DES-X este o variantă DES care rezistă mai bine (decât DES) la forța brută;
- DES-X folosește două chei suplimentare k_1, k_2:

$$DESX_{k,k_1,k_2} = k_2 \oplus DES_k(x \oplus k_1)$$

Important de reținut!

- DES a fost sistemul simetric dominant de la mijlocul anilor '70 până la mijlocul anilor '90;
- DES cu cheia pe 56 biți poate fi spart relativ ușor astăzi prin forță brută;
- Insă, este foarte greu de spart folosind criptanaliza diferențială sau liniară;
- Pentru 3DES nu se cunoaste nici un atac practic.

Securitatea Sistemelor Informatice

- Curs 7.1 -
Coduri de autentificare a mesajelor - MAC

Adela Georgescu

Facultatea de Matematică și Informatică
Universitatea din București
Anul universitar 2022-2023, semestrul I

Comunicare sigură și integritatea mesajelor

- Un scop de bază al criptografiei este să asigure comunicarea sigură de-a lungul unui canal public de comunicare;
- Am vazut cum putem obține confidențialitatea cu ajutorul schemelor de criptare;
- Insă, nu ne interesează doar ca adversarul să nu aibă acces la mesajele trimise, ci...
- Vrem să garantăm integritatea mesajelor (sau autentificarea mesajelor)
- Aceasta înseamnă ca mesajul primit de Bob este exact mesajul trimis de Alice.

Comunicare sigură și integritatea mesajelor

- Iată un exemplu:
- Să consideram cazul în care un mare lanț de supermarket-uri trimite o comandă pe email către un furnizor pentru a achiziționa 10.000 bax-uri de apă minerală;
- Odată primită comanda, furnizorul trebuie să verifice următoarele:
  1. Comanda este autentică? A fost trimisă cu adevărat de un supermarket sau de către un adversar care a furat contul de email al clientului respectiv ?
  2. Dacă s-a convins de autenticitatea comenzii, trebuie verificat dacă detaliile ei sunt cele originale sau au fost modificate pe parcurs de un adversar.

Comunicare sigură și integritatea mesajelor

- In exemplul precedent, problema este doar de integritate a mesajelor, și nu de confidențialitate (comanda nu e secretă);
- In general nu ne putem baza pe încredere în ceea ce privește integritatea mesajelor transmise, indiferent că ele sunt:
  - comenzi efectuate online
  - operațiuni bancare online
  - email, SMS
- Vom vedea cum putem folosi tehnici criptografice pentru a preveni modificarea nedectată a mesajelor transmise.

Criptare vs. autentificarea mesajelor

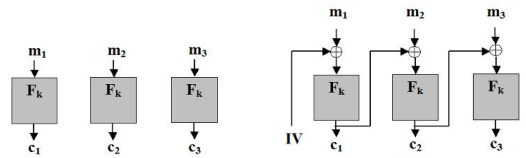
- Criptarea, în general, **NU** oferă integritatea mesajelor!
- Nu folositi criptarea cu scopul de a obtine autentificarea mesajelor
- Dacă un mesaj este transmis criptat de-a lungul unui canal de comunicare, nu înseamnă că un adversar nu poate modifica/altera mesajul așa încât modificarea să aiba sens în textul clar;
- Verificăm, în continuare, că nici o schemă de criptare studiată nu oferă integritatea mesajelor;

Criptare vs. autentificarea mesajelor

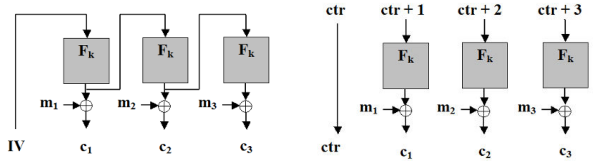
- Criptarea folosind sisteme fluide
  - $Enc_k(m) = G(k) \oplus m$, unde G este un PRG;
  - Dacă modificăm un singur bit din textul criptat c, modificarea se va reflecta imediat în același bit din textul clar;
  - Consecințele pot fi grave: de pildă, să considerăm transferul unei sume de bani în dolari criptate, reprezentată în binar;
  - Modificarea unui bit poate schimba suma foarte mult (al 11 lsb schimbă suma cu mai mult de 1000$);
  - Același atac se poate aplica și la OTP.

Criptare vs. autentificarea mesajelor

- Criptarea folosind sisteme bloc
  - Intrebare: Atacul de mai sus se poate aplica și pentru sistemele bloc cu modurile de operare studiate?



(a) ECB (b) CBC



(c) OFB (d) CTR

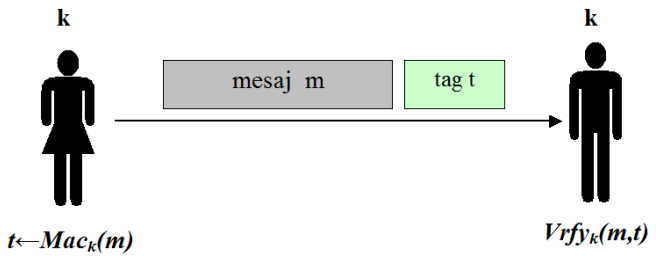
Criptare vs. autentificarea mesajelor

- Răspuns: Atacul se aplică identic pentru modurile OFB și CTR;
- Pentru modul ECB, modificarea unui bit din al i-lea bloc criptat afectează numai al i-lea bloc clar, dar este foarte greu de prezis efectul exact;
- Mai mult, ordinea blocurilor la ECB poate fi schimbată;
- Pentru modul CBC, schimbarea bitului j din IV va schimba bitul j din primul bloc;
- Toate celelalte blocuri de text clar rămân neschimbate ($m_i = F_k^{-1}(c_i) \oplus c_{i-1}$ iar blocurile c_i si c_{i-1} nu au fost modificate).

Coduri de autentificare a mesajelor - MAC

- Așa cum am vazut, criptarea nu rezolvă problema autentificarii mesajelor;
- Vom folosi un mecanism diferit, numit cod de autentificare a mesajelor - MAC (Message Authentication Code);
- Scopul lor este de a împiedica un adversar să modifice un mesaj trimis fără ca părțile care comunică să nu detecteze modificarea;
- Vom lucra în continuare în contextul criptografiei cu cheie secretă unde părțile trebuie să prestabilească de comun acord o cheie secretă.

MAC - Definiție



- Alice și Bob stabliesc o cheie secretă k pe care o partajează;
- Când Alice vrea să îi trimită un mesaj m lui Bob, calculează mai întâi un `tag` t (o etichetă) pe baza mesajului m și a cheii k și trimite perechea (m, t);

MAC - Definiție

- Tag-ul este calculat folosind un algoritm de generare a tag-urilor numit `Mac`;
- La primirea perechii (m, t) Bob verifică dacă tag-ul este valid (în raport cu cheia k) folosind un algoritm de verificare `Vrfy`;
- In continuare prezentăm definiția formală a unui cod de autentificare a mesajelor.

MAC - Definiție

Definiție

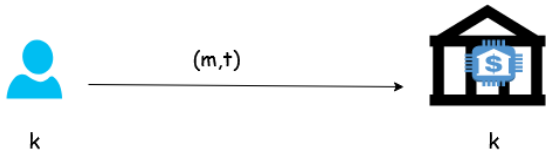
Un cod de autentificare a mesajelor (MAC) definit peste $(\mathcal{K}, \mathcal{M}, \mathcal{T})$ *este format dintr-un triplet de algoritmi polinomiali* $(\mathrm{Gen}, \mathrm{Mac}, \mathrm{Vrfy})$ *unde:*

1. $\mathrm{Gen}(1^n)$*: este algoritmul de generare a cheii* k *(aleasă uniform pe* n *biți)*
2. $\mathrm{Mac} : \mathcal{K} \times \mathcal{M} \to \mathcal{T}$ *este algoritmul de generare a tag-urilor* $t \leftarrow \mathrm{Mac}_k(m)$*;*
3. $\mathrm{Vrfy} : \mathcal{K} \times \mathcal{M} \times \mathcal{T} \to \{0,1\}$
   este algoritmul de verificare ce întoarce un bit $b = \mathrm{Vrfy}_k(m,t)$ *cu semnificația că:*
   - $b = 1$ *înseamnă* valid
   - $b = 0$ *înseamnă* invalid

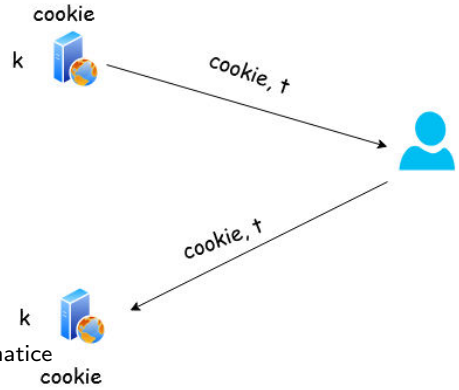
a.î : $\forall m \in \mathcal{M},\ k \in \mathcal{K}\ \mathrm{Vrfy}_k(m, \mathrm{Mac}_k(m)) = 1$.

Cazuri de folosire ale MAC

1. când cele două părti care comunică partajează o cheie secretă în avans



2. când avem o singură parte care autentifică o comunicație, interacționând cu ea însăși după un timp



Securitate MAC - discuție

- Intuiție: nici un adversar polinomial nu ar trebui să poată genera un tag valid pentru nici un mesaj "nou" care nu a fost deja trimis (și autentificat) de părțile care comunică;
- Trebuie să definim puterea adversarului și ce înseamnă spargerea sau un atac asupra securității;
- Adversarul lucrează în timp polinomial și are acces la mesajele trimise între părți împreuna cu tag-urile aferente.
- Adversarul este *activ*, poate influența autentificarea unor mesaje alese de el...
- ...dar nu trebuie să poată falsifica tag-ul aferent unor mesaje care nu au fost autentificate de expeditor

Securitate MAC - formalizare

- Formal, îi dăm adversarului acces la un *oracol* $\mathrm{Mac}_k(\cdot)$;
- Adversarul poate trimite orice mesaj m dorit către oracol și primește înapoi un tag corespunzător $t \leftarrow \mathrm{Mac}_k(m)$;
- Considerăm că securitatea este impactată dacă adversarul este capabil să producă un mesaj m împreună cu un tag t așa încât:
  1. t este un tag valid pentru mesajul m: $\mathrm{Vrfy}_k(m,t) = 1$;
  2. Adversarul nu a solicitat anterior (de la oracol) un tag pentru mesajul m.

Securitate MAC - formalizare

- Despre un MAC care satisface nivelul de securitate de mai sus spunem că *nu poate fi falsificat printr-un atac cu mesaj ales*;
- Aceasta înseamnă că un adversar nu este capabil să falsifice un tag valid pentru nici un mesaj ...
- ... deși poate obține tag-uri pentru orice mesaj ales de el, chiar *adaptiv* în timpul atacului.
- Pentru a da definiția formală, definim mai întâi un experiment pentru un MAC $\pi = (\mathrm{Mac}, \mathrm{Vrfy})$, în care considerăm un adversar $\mathcal{A}$ și parametrul de securitate n;

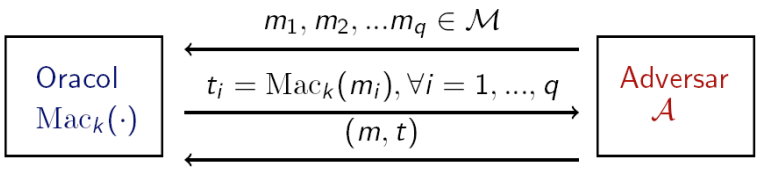
Securitate MAC

Definiție

Un cod de autentificare al mesajelor $\pi = (\mathrm{Gen}, \mathrm{Mac}, \mathrm{Vrfy})$ este sigur (nu poate fi falsificat printr-un atac cu mesaj ales) dacă pentru orice adversar polinomial $\mathcal{A}$ există o funcție neglijabilă negl așa încât

$$Pr[\mathrm{Mac}^{forge}_{\mathcal{A},\pi}(n) = 1] \leq negl(n).$$

Experimentul $\mathrm{Mac}^{forge}_{\mathcal{A},\pi}(n)$



- Output-ul experimentului este 1 dacă și numai dacă: (1) $\mathrm{Vrfy}_k(m, t) = 1$ si (2) $m \notin \{m_1, ..., m_q\}$;
- Dacă $\mathrm{Mac}^{forge}_{\mathcal{A},\pi}(n) = 1$, spunem că $\mathcal{A}$ a efectuat experimentul cu succes.

Atacuri prin replicare

- Întrebare: De ce este necesară a doua condiție de la securitatea MAC (un adversar nu poate întoarce un mesaj pentru care anterior a cerut un tag)?
- Răspuns: Pentru a evita atacurile triviale în care un adversar cere tag-ul aferent unui mesaj și apoi întoarce chiar acel mesaj împreună cu tag-ul primit.
- Intrebare: Definiția MAC oferă protecție la atacurile prin replicare (in care un adversar copiază un mesaj împreună cu tag-ul aferent trimise de părțile comunicante)?
- Răspuns: NU! MAC-urile nu oferă nici un fel de protecție la atacurile prin replicare.

Atacuri prin replicare

- De exemplu: Alice trimite către banca sa un ordin de transfer a 1.000$ din contul ei în contul lui Bob;
- Pentru aceasta, Alice calculează un tag MAC și îl atașază mesajului așa încât banca știe că mesajul este autentic;
- Dacă MAC-ul este sigur, Bob nu va putea intercepta mesajul și modifica suma la 10.000$;
- Dar Bob poate intercepta mesajul și îl poate replica de *zece* ori către bancă;
- Dacă banca îl acceptă, Bob va avea în cont 10.000$.

Atacuri prin replicare

- Un MAC nu protejează împotriva unui atac prin replicare pentru că definiția nu încorporează nici o noțiune de *stare* în algortimul de verificare;
- Mai degrabă, protecția împotriva replicării trebuie făcută la nivel înalt de către aplicatiile care folosesc MAC-uri;

Construcția MAC-urilor sigure

- Avem nevoie de o funcție cu cheie, pentru care, dându-se $Mac_k(m_1), Mac_k(m_2)...$
- ... să nu fie ușor (în timp polinomial) a găsi $Mac_k(m)$ pentru orice $m \notin \{m_1, m_2, ...\}$
- Funcția MAC ar putea fi un PRF

Construcția MAC-urilor sigure

- *Funcțiile pseudoaleatoare* (PRF) sunt un instrument bun pentru a construi MAC-uri sigure;

Construcție

Fie $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ o PRF. Definim un MAC în felul următor:

- Mac : *pentru o cheie $k \in \{0,1\}^n$ și un mesaj $m \in \{0,1\}^n$, calculează tag-ul $t = F_k(m)$ (dacă $|m| \neq |k|$ nu întoarce nimic);*
- Vrfy : *pentru o cheie $k \in \{0,1\}^n$, un mesaj $m \in \{0,1\}^n$ și un tag $t \in \{0,1\}^n$, întoarce 1 dacă și numai dacă $t = F_k(m)$ (dacă $|m| \neq |k|$, întoarce 0).*

Construcția MAC-urilor sigure

Teoremă

Dacă F este PRF, construcția de mai sus reprezintă un cod de autentificare a mesajelor sigur (nu poate fi falsificat prin atacuri cu mesaj ales).

MAC-uri pentru mesaje de lungime variabilă

- Construcția prezentată anterior funcționează doar pe mesaje de lungime fixă (PRF-urile sunt instanțiate cu sisteme de criptare bloc care, cel mai adesea, acceptă input-uri de 128 biți);
- Insă în practică avem nevoie de mesaje de lungime variabilă;
- Arătăm cum putem obține un MAC de lungime variabilă pornind de la un MAC de lungime fixă;
- Fie $(\pi' = (\mathrm{Mac}', \mathrm{Vrfy}'))$ un MAC sigur de lungime fixă pentru mesaje de lungime n;
- Pentru a construi un MAC de lungime variabilă, putem sparge mesajul m în blocuri $m_1, ..., m_d$ și autentificam blocurile folosind π';
- Iată câteva modalități de a face aceasta:

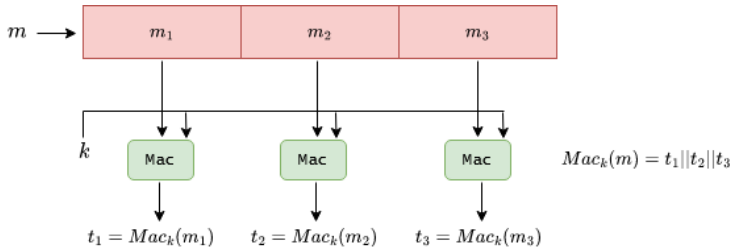
MAC-uri pentru mesaje de lungime variabilă

1. *XOR pe toate blocurile cu autentificarea rezultatului*:
$$t = \mathrm{Mac}'_k(\oplus_i m_i)$$

- Intrebare: Este sigură această metodă?
- Răspuns: NU! Un adversar poate modifica mesajul original m a.î. XOR-ul blocurilor nu se schimbă, el obținând un tag valid pentru un mesaj nou;

MAC-uri pentru mesaje de lungime variabilă

2. *Autentificare separată pentru fiecare bloc*:

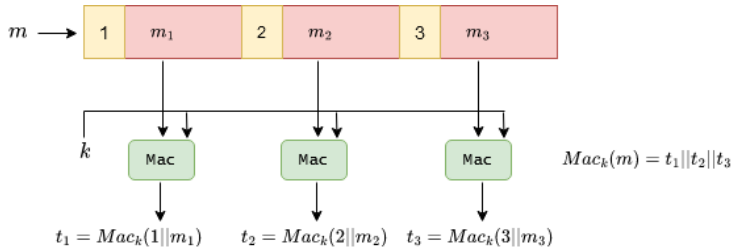


- Intrebare: Este sigură această metodă?
- Răspuns: NU!
- Fiind dat (m, t) cu $m = m_1||m_2||m_3$ și $t = t_1||t_2||t_3$
- Atunci (m', t') este o pereche validă cu $m' = m_2||m_1||m_3$ și $t' = t_2||t_1||t_3$

MAC-uri pentru mesaje de lungime variabilă

3. *Autentificare separată pentru fiecare bloc folosind o secvență de numere*:

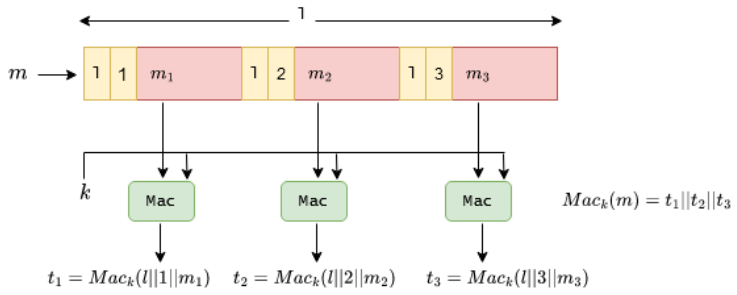
In felul acesta prevenim atacul anterior



- Intrebare: Este sigură această metodă?
- Răspuns: NU! Un adversar poate scoate blocuri de la sfârșitul mesajului: $t = t_1||t_2$ este un tag valid pentru mesajul $m = m_1||m_2$;

MAC-uri pentru mesaje de lungime variabilă

- Soluție pentru atacurile anterioare: adăugarea de informație suplimentară în fiecare bloc



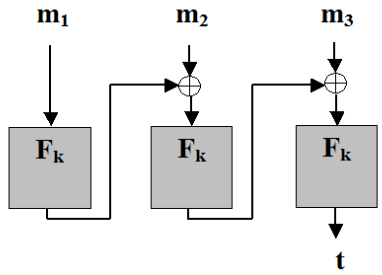
- Intrebare: Este sigură această metodă?
- Răspuns: NU!
- Pentru perechea (m, t) cu $m = m_1||m_2||m_3$ și $t = t_1||t_2||t_3$ și perechea (m', t') cu $m' = m'_1||m'_2||m'_3$ și $t' = t'_1||t'_2||t'_3$ iar $|m| = |m'|$
- perechea $(m_1||m'_2||m_3, t_1||t'_2||t_3)$ este una validă

CBC-MAC

- Soluția este ineficientă și greu de folosit în practică;
- Insă, am văzut că putem construi MAC-uri sigure (chiar pentru mesaje de lungime variabilă) pe baza funcțiilor pseudoaleatoare (intrare de lungime fixă);
- Ceea ce înseamnă că putem construi MAC-uri sigure pornind de la cifruri bloc;
- Dar, cu construcția de mai sus, rezultatul e foarte ineficient: pentru un tag aferent unui mesaj de lungime $l \cdot n$, trebuie să aplicăm sistemul bloc de $4l$ ori iar tag-ul rezultat are $(4l + 1)n$ biți;

CBC-MAC

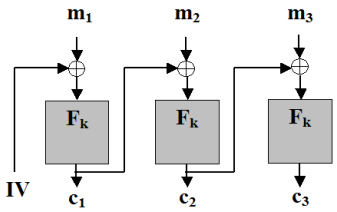
- O soluție mult mai eficientă este să folosim CBC-MAC;
- CBC-MAC este o construcție similară cu modul CBC folosit pentru criptare;
- Folosind CBC-MAC, pentru un tag aferent unui mesaj de lungime $l \cdot n$, se aplică sistemul bloc doar de l ori, iar tag-ul are numai n biți.



Securitatea CBC-MAC

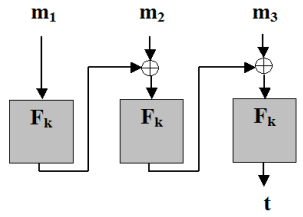
- Dacă F este o funcție pseudoaleatoare, construcția de mai sus reprezintă un cod de autentificare a mesajelor sigur (nu poate fi falsificat prin atacuri cu mesaj ales) pentru mesaje de lungime $\ell \cdot n$ pentru ℓ fixat.
- Construcția prezentată este sigură numai pentru autentificarea mesajelor de lungime fixă;
- ℓ fixat înseamnă că cele două părți care comunică trebuie să partajeze ℓ în avans
- Avantajul acestei construcții față de cea anterioară este că ea poate autentifica mesaje de lungime mult mai mare;

CBC-MAC vc. Criptare în modul CBC



Criptare în mod CBC

- IV este aleator pentru a obține securitate;
- toate blocurile c_i constituie mesajul criptat.



CBC-MAC

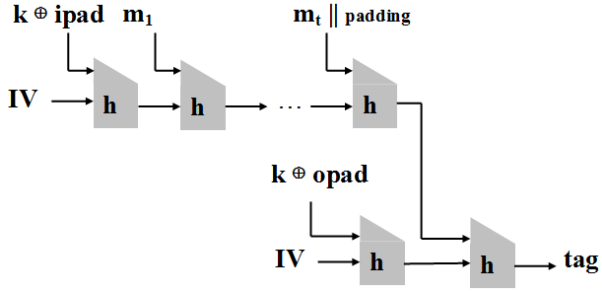
- $IV = 0^n$ este fixat pentru a obține securitate;
- doar ieșirea ultimului bloc constituie tag-ul (întoarcerea tuturor blocurilor intermediare duce la pierderea securității)

Construcții MAC

- Există două construcții de bază care se folosesc în practică:
  - CBC-MAC - folosit pe larg în industria bancară
  - HMAC - pentru protocoale pe Internet: SSL, IPsec, SSH...
- Am studiat CBC-MAC mai devreme în curs;
  In continuare trecem în revistă HMAC

HMAC

- Folosim metoda standardizată HMAC (Hash MAC):



- HMAC se definește astfel:
- $\mathrm{Mac}(k, m)$: $t = H((k \oplus \mathtt{opad}) \,||\, H((k \oplus \mathtt{ipad}) \,||\, m))$;
- $\mathrm{Vrfy}(k, m, t) = 1 \iff t = \mathrm{Mac}(k, m)$.

Notații

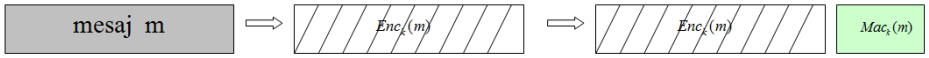
- `ipad` și opad sunt două constante de lungimea unui bloc m_i
- `ipad` constă din byte-ul $0x5C$ repetat de atâtea ori cât e nevoie;
- opad constă din byte-ul $0x36$ repetat de atâtea ori cât e nevoie;
- IV este o constantă fixată.

Confidențialitate și integritate - criptare autentificată

- Am văzut cum putem obține confidențialitate folosind scheme de criptare;
- Am văzut cum putem garanta integritatea datelor folosind MAC-uri;
- In practică avem nevoie de ambele proprietăți de securitate: confidențialitate și integritatea datelor adica criptare autentificată - authenticated encryption
- Nu orice combinație de schemă de criptare sigură și MAC sigur oferă cele două proprietăți de securitate!

Confidențialitate și integritate - criptare autentificată

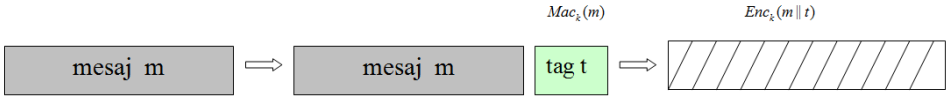
- Iată trei abordari uzuale pentru a combina criptarea și autentificarea mesajelor:
  1. *Criptare-și-autentificare*: criptarea și autentifcarea se fac independent. Pentru un mesaj clar m, se transmite mesajul criptat $\langle c, t \rangle$ unde
     $$c \leftarrow \mathrm{Enc}_{k_1}(m) \text{ și } t \leftarrow \mathrm{Mac}_{k_2}(m)$$
     La recepție, $m = \mathrm{Dec}_{k_1}(c)$ și dacă $\mathrm{Vrfy}_{k_2}(m, t) = 1$, atunci întoarce m; altfel întoarce $\perp$.



- Combinația aceasta **nu** este neaparat sigură; un MAC sigur nu implică nici un fel de confidențialitate;

Confidențialitate și integritate - criptare autentificată

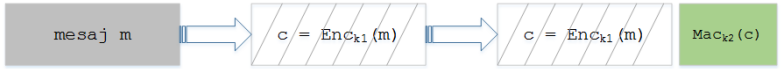
- 2. *Autentificare-apoi-criptare*: întâi se calculează tag-ul t apoi mesajul și tag-ul sunt criptate împreună
  $$t \leftarrow \mathrm{Mac}_{k_2}(m) \text{ si } c \leftarrow \mathrm{Enc}_{k_1}(m||t)$$
  La recepție, $m||t = \mathrm{Dec}_{k_1}(c)$ și dacă $\mathrm{Vrfy}_{k_2}(m, t) = 1$, atunci întoarce m; altfel întoarce $\perp$.



- Combinația aceasta **nu** este neapărat sigură;
- Se poate construi o schemă de criptare CPA-sigură care împreună cu orice MAC sigur nu poate fi CCA-sigură;

Securitate Criptare-apoi-autentificare

3. *Criptare-apoi-autentificare*



- Combinația aceasta este întotdeauna sigură; se folosește în IPsec;
- Deși folosim aceeași construcție pentru a obține securitate CCA și transmitere sigură a mesajelor, scopurile urmărite sunt diferite în fiecare caz;

Necesitatea de a folosi chei diferite

- Pentru scopuri diferite de securitate trebuie să folosim întotdeauna chei diferite;
- Să urmărim ce se întâmplă dacă folosim metoda criptare-apoi-autentificare cu aceeași cheie k atât pentru criptare cât și pentru autentificare;
- Definim $\mathrm{Enc}_k(m) = F_k(m||r)$, pentru $m \in \{0,1\}^{n/2}$, $r \leftarrow \{0,1\}^{n/2}$ aleator, iar $F_k(\cdot)$ o permutare pseudoaleatoare;
- Definim $\mathrm{Mac}_k(c) = F_k^{-1}(c)$;
- Schema de criptare și MAC-ul sunt sigure dar...
- $\mathrm{Enc}_k(m), \mathrm{Mac}_k(\mathrm{Enc}_k(m)) = F_k(m||r), F_k^{-1}(F_k(m||r)) = F_k(m||r), m||r.$

Securitate CCA vs. Criptare autentificată

- Orice schemă de criptare autentificată este și CCA-sigură dar există și scheme de criptare CCA-sigure care nu sunt scheme de criptare autentificată
- Totusi, cele mai multe aplicații de scheme de criptare cu cheie secretă în prezența unui adversar activ necesită integritate

Criptare autentificată in practică

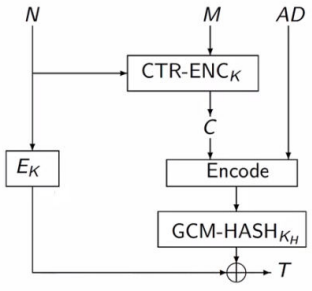
CA in	bazata pe	care in general este	dar in acest caz este
SSH	C_si_A	nesigura	sigura
SSL	A_apoi_C	nesigura	nesigura
IPSec	C_apoi_A	sigura	sigura
WinZip	C_apoi_A	sigura	nesigura

- CA = criptare autentificată;
  C-apoi-A = criptare-apoi-autentificare;
  C-si-A = criptare-si-autentificare

Criptare autentificată in practică

- Scheme de criptare autentificată: OCB, GCM, CCM, EAX
- TLS folosește GCM
- Competiția CAESAR pentru standardizarea de noi scheme de criptare autentificată
  http://competitions.cr.yp.to/caesar.html

Exemplu: GCM



- CTR-ENC reprezintă criptare în modul CTR
- GCM-HASH$_{K_H}$ - funcție hash bazată pe polinoame (peste corp Galois binar)
- K_H este derivată din E și K

Securitatea Sistemelor Informatice

- Curs 7.2 -
Funcții hash

Adela Georgescu

Facultatea de Matematică și Informatică
Universitatea din București
Anul universitar 2022-2023, semestrul I

Funcții Hash

- Întrebare: Ați auzit vreodată de funcții hash? În ce context?
- Acestea primesc ca argument o secvență de *lungime variabilă* pe care o comprimă într-o secvență de *lungime mai mică, fixată*;
- Utilizarea clasică a funcțiilor hash este în domeniul *structurilor de date*;
- Să luăm un exemplu...

Funcţii Hash

- Considerăm o mulţime de elemente de dimensiune mare care trebuie stocată (într-un tablou);

Adresă
Batiştei nr.17
Academiei nr.23
Tudor Arghezi nr.103
Nicolae Bălcescu nr.10
C.A.Rosetti nr.7
Hristo Botev nr.35
...

- Ulterior, elementele trebuie să fie uşor accesibile;
- Întrebare: Cum procedăm?

Funcţii Hash

- **Varianta 1.** Elementele se stochează la rând;

Index	Adresă
1	Batiştei nr.17
2	Academiei nr.23
3	Tudor Arghezi nr.103
4	Nicolae Bălcescu nr.10
5	C.A.Rosetti nr.7
6	Hristo Botev nr.35
...	... ...

- Dar o căutare necesită un algoritm de complexitate ... $O(n)$;

Funcţii Hash

- **Varianta 2.** Tabloul de elemente este sortat.

Index	Adresă
...	... ...
17	Academiei nr.23
...	... ...
120	Batiştei nr.17
...	... ...
223	C.A.Rosetti nr.7
...	... ...
401	Hristo Botev nr.35
...	... ...
503	Nicolae Bălcescu nr.10
...	... ...
696	Tudor Arghezi nr.103
...	... ...

- În acest caz o căutare necesită un algoritm de complexitate ... $O(\log n)$;

Funcţii Hash

- **Varianta 3.** Se foloseşte o funcţie hash H şi fiecare element x se stochează la indexul $H(x)$;

Index	Adresă
...	... ...
14	Tudor Arghezi nr.103
...	... ...
29	Batiştei nr.17
...	... ...
113	C.A.Rosetti nr.7
...	... ...
365	Academiei nr.23
...	... ...
411	Nicolae Bălcescu nr.10
...	... ...
703	Hristo Botev nr.35
...	... ...

- Căutarea devine optimă pentru că se realizează în ... $O(1)$!

Funcții Hash

- În exemplul precedent:
  - H("Tudor Arghezi nr.103") = 14;
  - H("Batiștei nr.17") = 29;
  - ...
- Analizăm, pe rând, cele 3 operații: **căutare**, **adăugare**, **ștergere**;
- Pentru **căutarea** adresei *Edgar Quinet nr.35*, se calculează H("Edgar Quinet nr.35");
- Presupunând că H("Edgar Quinet nr.35") = 79, atunci se verifică ce este stocat pe poziția 79;

Funcții Hash

- Pentru **introducerea** adresei *Nicolae Filipescu nr.31*, se calculează H("Nicolae Filipescu nr.31");
- Presupunând că H("Nicolae Filipescu nr.31") = 153, atunci se introduce valoarea *Nicolae Filipescu nr.31* la indexul 153;
- Pentru **ștergerea** adresei *Hristo Botev nr.35*, se calculează H("Hristo Botev nr.35");
- Se obține H("Hristo Botev nr.35") = 401 și se eliberează această celulă;

Funcții Hash

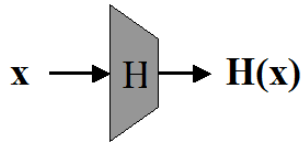
- Întrebare: Ce se întâmplă dacă pentru 2 valori $x \neq x'$, $H(x) = H(x')$?
- Răspuns: În acest caz, apare o coliziune - ambele valori se vor stoca în aceeași celulă.

Funcții Hash

- În criptografie, o funcție hash rămâne o funcție care comprimă secvențe de lungime variabilă în secvențe de lungime fixă, insă:
- Dacă în contextul structurilor de date este *preferabil* să se minimizeze coliziunile, în criptografie acest lucru este **impus**;
- Dacă în contextul structurilor de date coliziunile apar *arbitrar* (valorile sunt alese independent de funcția hash), în criptografie coliziunile trebuie evitate chiar dacă sunt căutate **în mod voit** (de către adversar).

Funcții Hash

- Întrebare: Există funcții hash fără coliziuni?
- Răspuns: NU! Funcțiile hash (cel puțin cele interesante d.p.d.v criptografic) comprimă, deci funcția nu poate fi injectivă;



Funcții Hash

- În aceste condiții, o funcție hash impune ca determinarea unei coliziuni să devină dificilă;
- Considerăm funcțiile hash de domeniu infinit și ieșire de lungime fixată $l(n)$, unde $l(n)$ este un polinom în parametrul de securitate n:

$$H : \{0,1\}^* \rightarrow \{0,1\}^{l(n)}$$

Experimentul $Hash^{coll}_{\mathcal{A},H}(n)$

- Considerăm experimentul $Hash^{coll}_{\mathcal{A},H}(n)$;
- Adversarul $\mathcal{A}$ indică 2 valori $x, x' \in \{0,1\}^*$;
- Se definește valoarea experimentului $Hash^{coll}_{\mathcal{A},H}(n) = \mathbf{1}$ dacă $x \neq x'$ și $H(x) = H(x')$;
- Altfel, $Hash^{coll}_{\mathcal{A},H}(n) = \mathbf{0}$.

Rezistența la coliziuni

Definiție

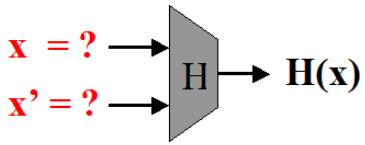
$H : \{0,1\}^* \rightarrow \{0,1\}^{l(n)}$ se numește rezistentă la coliziuni (collision-resistant) dacă pentru orice adversar PPT $\mathcal{A}$ există o funcție neglijabilă negl așa încât

$$Pr[Hash^{coll}_{\mathcal{A},H}(n) = 1] \leq negl(n).$$

Securitatea funcțiilor hash

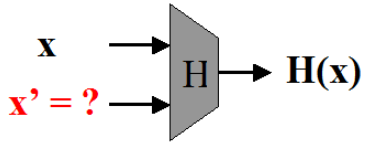
- În practică, rezistența la coliziuni poate fi dificil de obținut;
- Pentru anumite aplicații sunt utile noțiuni mai relaxate de securitate;
- Există 3 nivele de securitate:
  1. Rezistența la coliziuni: este cea mai puternică noțiune de securitate și deja am definit-o formal;
  2. Rezistența la a doua preimagine: presupune că fiind dat x este dificil de determinat $x' \neq x$ a.î. $H(x) = H(x')$
  3. Rezistența la prima preimagine: presupune că fiind dat $H(x)$ este imposibil de determinat x.

Rezistența la coliziuni



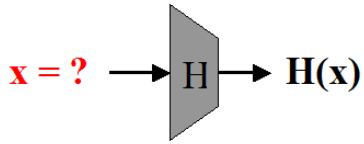
- **Provocare:** Se cer 2 valori $x \neq x'$ a.î. $H(x) = H(x')$;
- **Atac**: "Atacul zilei de naștere" necesită $\approx 2^{l(n)/2}$ evaluări pentru H.

Rezistența la a doua preimagine



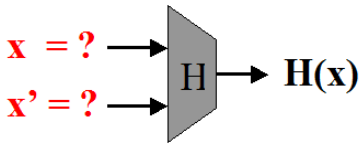
- **Provocare:** Fiind dat x, se cere x' a.î. $H(x) = H(x')$;
- **Atac**: Un atac generic necesită $\approx 2^{l(n)}$ evaluări pentru H.

Rezistența la prima preimagine



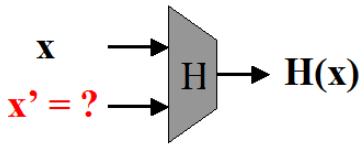
- **Provocare:** Fiind dat $y = H(x)$, se cere x a.î. $H(x) = y$;
- O astfel de funcție se numește și *calculabilă într-un singur sens (one-way function)*;
- **Atac**: Un atac generic necesită $\approx 2^{l(n)}$ evaluări pentru H.

Atacuri în practică

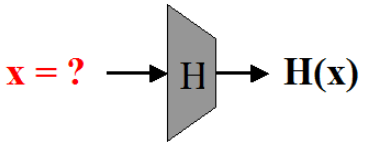


$\{x\}$ = documente originale
$\{x'\}$ = documente false
cineva este de acord să semneze electronic documentul original, însă semnătura devine valabilă și pentru un document fals

---



documentul x care este semnat electronic poate fi înlocuit de documentul x'



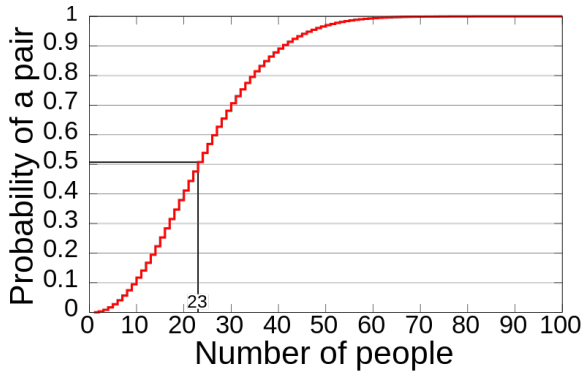
dacă se cunoaște cheia de sesiune k_i calculată din cheia master k
$x = H(k||i)$, atunci se determină cheia k

Securitatea funcțiilor hash

- Întrebare: De ce o funcție care satisface proprietatea de **rezistență la coliziuni** satisface și proprietatea de **rezistență la a doua preimagine**?
- Răspuns: Pentru x fixat, adversarul determină $x' \neq x$ pentru care $H(x) = H(x')$, deci găsește o coliziune;
- Întrebare: De ce o funcție care satisface proprietatea de **rezistență la a doua preimagine** satisface și proprietatea de **rezistență la prima imagine**?
- Răspuns: Pentru x oarecare, adversarul calculează $H(x)$, o inversează și determină x' a.î. $H(x') = H(x)$. Cu probabilitate mare $x' \neq x$, deci găsește o a doua preimagine.

Atacul "zilei de naștere"

- Analizăm posibilitatea de a determina o coliziune pornind de la un exemplu clasic: *paradoxul nașterilor*;
- Întrebare: Care este dimensiunea unui grup de oameni pentru ca 2 dintre ei să fie născuți în aceeași zi cu probabilitate 1/2 ?
- Răspuns: 23!



[Wikipedia]

Atacul "zilei de naștere"

- Generalizând, considerăm o mulțime de dimensiune n și q elemente uniform aleatoare din această mulțime $y_1, \ldots, y_q$;
- Atunci pentru $q \geq 1.2 \times 2^{n/2}$ probabilitatea să existe $i \neq j$ a.î. $y_i = y_j$ este $\geq 1/2$.
- Aceast rezultat conduce imediat la un atac asupra funcțiilor hash cu scopul de a determina coliziuni:
  - Adversarul alege $2^{n/2}$ valori x_i;
  - Calculează pentru fiecare $y_i = H(x_i)$;
  - Caută $i \neq j$ cu $H(x_i) = H(x_j)$;
  - Dacă nu găsește nici o coliziune, reia atacul.
- Cum probabilitatea de succes a atacului este $\geq 1/2$, atunci numărul de încercări este ≈ 2.

Transformarea Merkle-Damgård

- Numim funcție de compresie o funcție hash rezistentă la coliziuni de intrare de lungime fixă;
- Întrebare: Intuitiv, ce se construiește mai ușor, H_1 sau H_2 ?

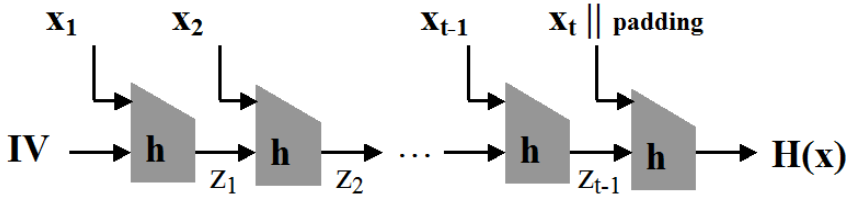
$$H_1 : \{0,1\}^{m_1} \to \{0,1\}^{n_1}, m_1 > n_1, m_1 \approx n_1$$

$$H_2 : \{0,1\}^{m_2} \to \{0,1\}^{n_2}, m_2 \gg n_2$$

- Răspuns: Cu cât domeniul și codomeniul funcției sunt mai apropiate ca dimensiune, numărul de coliziuni scade $\Rightarrow$ coliziunile devin mai dificil de determinat (considerăm că funcția distribuie valorile uniform aleator).

Transformarea Merkle-Damgård

- Scopul este să construim o funcție hash (cu intrare de lungime variabilă), pornind de la o funcție de compresie (de lungime fixă);
- Pentru aceasta se aplică transformarea Merkle-Damgård;



Notații

- h = o funcție de compresie de dimensiune fixă
- $x = x_1||x_2||\ldots||x_{t-1}||x_t$ = valoarea de intrare
- IV = vector de inițializare
- padding = 100...0|| lungimea mesajului

Transformarea Merkle-Damgård

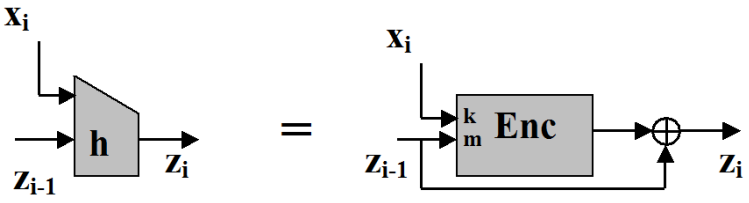
Teoremă

Dacă h prezintă rezistență la coliziuni, atunci și H prezintă rezistență la coliziuni.

- Intuitiv, dacă există $x \neq x'$ a.î. $H(x) = H(x')$, atunci trebuie să existe $< z_{i-1}, x_i > \neq < z'_{i-1}, x'_i >$ a.î. $h(z_{i-1}||x_i) = h(z'_{i-1}||x'_i)$;
- Altfel spus, dacă se găsește o coliziune pentru H se găsește o coliziune și pentru h.

Transformarea Merkle-Damgård

- Rămâne să prezentăm o construcție pentru h;
- Construcția Davies-Meyer:



- Enc este un sistem bloc care criptează z_{i-1} cu cheia x_i:

$$h(z_{i-1}||x_i) = Enc_{x_i}(z_{i-1}) \oplus z_{i-1}$$

Exemple

- MD5 (Message Digest 5)
  - definit în 1991 de R.Rivest ca înlocuitor pentru MD4
  - produce o secvență hash de 128 biți
  - nesigur din 1996
  - utilizat în versiuni mai vechi de Moodle
- SHA (Secure Hash Algorithm)
  - o familie de funcții hash publicate de NIST
  - SHA-0 și SHA-1 sunt nesigure
  - SHA-2 prezintă 2 variante: SHA-256 și SHA-512
  - SHA-3 adoptat în 2012 pe baza Keccak

Exercitii funcții hash

- Este $H(x) = x(mod\ N)$ o funcție hash rezistenta la coliziuni? Daca da, explicati de ce, daca nu, gasiți o coliziune.
- Raspuns: Funcția nu e rezistenta la coliziuni: pentru orice $x < N$ si orice $a \in \mathbb{N}$, x si $aN + x$ formeaza o coliziune.
- Este $H(x) = ax + b(mod\ N)$ cu $(a, N) = 1$ o functie hash rezistenta la coliziuni? Daca da, explicați de ce, daca nu, gasiți o coliziune.
- Raspuns: Funcția nu e rezistenta la coliziuni: cautam x_1 si x_2 asa incat $H(x_1) = H(x_2)$ adica $ax_1 + b = ax_2 + b(mod\ N)$. Obținem $a(x_1 - x_2) = 0(mod\ N)$. Cum (a,N) obținem ca $x_1 - x_2 = 0(mod\ N)$.

Securitatea Sistemelor Informatice

- Curs 8.1 -
SHA-3

Adela Georgescu

Facultatea de Matematică și Informatică
Universitatea din București
Anul universitar 2022-2023, semestrul I

Exemple de funcții hash

- MD5
  - output pe 128 biți
  - propusă în 1991 și considerată rezistentă la coliziuni o perioadă de timp
  - 2004 - atac pentru găsirea de coliziuni + coliziuni explicite
  - astăzi se pot găsi coliziuni în mai puțin de un minut pe un calculator desktop

Exemple de funcții hash

- SHA-1
  - face parte din seria de algoritmi standardizați SHA (Secure Hash Algorithm) si a fost standardul aprobat de NIST până în 2011
  - output pe 160 biti
  - 2005 - atac teoretic pentru găsirea coliziunilor; necesită 2^{69} evaluări ale funcției hash
  - 2017 - prima coliziune practică - 2^{63} evaluări ale funcției hash
  - atac pentru găsirea de *coliziuni cu prefix* - aprox. 2^{67} evaluari
  - *coliziuni cu prefix* - pornind de la prefixele P si P', se cere găsirea mesajelor $M \neq M'$ cu $H(P||M) = H(P'||M')$
  - în practică, acestea sunt mai periculoase, pot duce la diverse atacuri incluzând generarea de certificate digitale false și atacuri asupra TLS, SSH

Exemple de funcții hash

- SHA-2
  - prezintă două versiuni: SHA-256 și SHA-512 în funcție de lungimea output-ului
  - nu se cunosc vulnerabilități; atât SHA-2 cât și SHA-3 sunt sigur de folosit acolo unde rezistența la coliziuni este necesară

Competiția SHA-3

- Atacurile asupra MD5 si SHA-1 au impus necesitatea unei noi funcții hash;
- 2 noiembrie 2007 - NIST anunță competiția publică SHA-3;
- 31 octombrie 2008 - se primesc 64 de propuneri din toată lumea;
- decembrie 2008 - rămân 51 de candidați pentru prima rundă (restul sunt eliminați din cauza dosarelor incomplete);

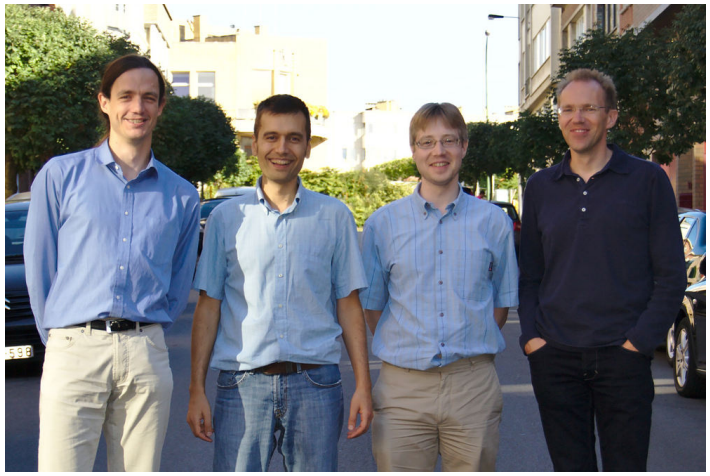
Competiția SHA-3

- februarie 2009 - are loc prima conferinta la care sunt prezentate 37 de propuneri (dintr-un total de 41, 10 fiind retrase între timp din cauza unor atacuri);
- iulie 2009 - rămân 14 candidați în runda a 2-a;
- decembrie 2010 - cei 5 candidați în runda finală: BLAKE, Grϕstl, JH, Keccak and Skein;
- 2 octombrie 2012 - NIST anunță câștigătorul: Keccak.

Cei 5 finaliști

BLAKE	Jean-Philippe Aumasson, Luca Henzen, Willi Meier, Raphael C.-W. Phan
Grϕstl	Lars Ramkilde Knudsen, Praveen Gauravaram, Krystian Matusiewicz, Florian Mendel, Christian Rechberger, Martin Schläffer, Søren S. Thomsen
JH	Hongjun Wu
Keccak	Joan Daemen, Guido Bertoni, Michaël Peeters, Gilles Van Assche
Skein	Bruce Schneier, Niels Ferguson, Stefan Lucks, Doug Whiting, Mihir Bellare, Tadayoshi Kohno, Jesse Walker, Jon Callas

Echipa Keccak



[http://keccak.noekeon.org/team.html]

Criterii de selecție

- Lungimea secvenței de ieșire: $n = 224, 256, 384$ și 512 biți;
- Alte dimensiuni ale secvenței de ieșire sunt opționale;
- Eficientă crescută față de SHA-2;
- Utilizare în HMAC;
- Rezistența la coliziuni, prima și a doua preimagine (conform cu atacurile generice, tradiționale);
- Demonstrație de securitate;
- Parametrizabilă, număr de runde variabil;
- Simplicitate, claritate.

Motivație

"The NIST team praised the Keccak algorithm for its many admirable qualities, including its elegant design and its ability to run well on many different computing devices. The clarity of Keccak's construction lends itself to easy analysis (during the competition all submitted algorithms were made available for public examination and criticism), and Keccak has higher performance in hardware implementations than SHA-2 or any of the other finalists."

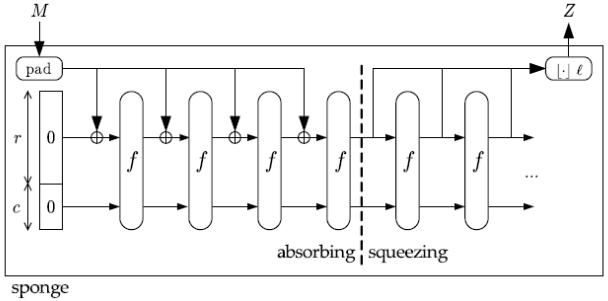
(NIST Selects Winner of Secure Hash Algorithm (SHA-3) Competition - `http://www.nist.gov/itl/csd/sha-100212.cfm`)

"One benefit that KECCAK offers as the SHA-3 winner is its difference in design and implementation properties from that of SHA-2. It seems very unlikely that a single new cryptanalytic attack or approach could threaten both algorithms."

(SHA-3 Selection Announcement - `http://csrc.nist.gov/groups/ST/hash/sha-3/sha-3_selection_announcement.pdf`)

Keccak

- A fost gândit să difere complet de construcțiile existente (AES, SHA-2);
- Folosește sponge functions:
- Principala componentă este permutarea f care acceptă blocuri de 1600 biți.



[Cryptographic Sponge Functions - `http://sponge.noekeon.org/CSF-0.1.pdf`]

Keccak

- Notații:
  - r = *bitrate*
  - c = *capacity*
  - $b = c + r$ = *width*
  - f = o permutare
- Folosește o stare de b biți inițializată la 0;
- Presupune 2 etape:
  1. Absorbing phase: mesajul de intrare se sparge în blocuri de lungime r care se XOR-ează la prima parte a stării, alternând cu aplicarea funcției f;
  2. Squeezing phase: partea superioară a stării este returnată la ieșire, alternând cu aplicarea funcției f; numărul de iterații depinde de numărul de biți l necesari la ieșire.

Keccak

Tabelul de mai jos din standardul NIST (https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf) ofera o comparatie d.p.d.v. al securitatii cu SHA-1 si SHA-2.

Function	Output Size	Security Strengths in Bits		
		Collision	Preimage	2nd Preimage
SHA-1	160	< 80	160	$160 - L(M)$
SHA-224	224	112	224	$\min(224, 256 - L(M))$
SHA-512/224	224	112	224	224
SHA-256	256	128	256	$256 - L(M)$
SHA-512/256	256	128	256	256
SHA-384	384	192	384	384
SHA-512	512	256	512	$512 - L(M)$
SHA3-224	224	112	224	224
SHA3-256	256	128	256	256
SHA3-384	384	192	384	384
SHA3-512	512	256	512	512
SHAKE128	d	$\min(d/2, 128)$	$\geq \min(d, 128)$	$\min(d, 128)$
SHAKE256	d	$\min(d/2, 256)$	$\geq \min(d, 256)$	$\min(d, 256)$

Important de reținut!

- Keccak este câștigătorul competiției SHA-3
- SHA-2 rămâne în continuare sigură

Securitatea Sistemelor Informatice

- Curs 8.2 -
Aplicații ale funcțiilor hash

Adela Georgescu

Facultatea de Matematică și Informatică
Universitatea din București
Anul universitar 2022-2023, semestrul I

Atac pentru găsirea de coliziuni

- Considerăm funcții hash

$$H : \{0,1\}^* \to \{0,1\}^n$$

- Rețineți! Cel mai bun atac generic pentru găsirea de coliziuni ne spune că avem nevoie de functii hash cu output-ul pe $2n$ biți pentru securitate împotriva adversarilor care rulează în timp 2^n...
- ... spre deosebire de sistemele de criptare bloc, unde avem nevoie de n biți pentru securitate împotriva adversarilor care rulează în timp 2^n
- Exemplu. Dacă dorim ca găsirea de coliziuni să necesite timp 2^{128} atunci trebuie să alegem functii hash cu output-ul pe 256 biți

Alte aplicații ale funcțiilor hash

- Am vazut ca functiile hash pot fi folosite pentru a construi MAC-uri de lungime variabila (în HMAC)

$$x \to H(x) \to Mac_k(H(x))$$

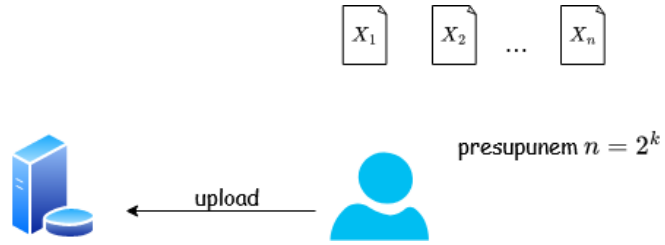
  pentru MAC de lungime fixă
- Functiile hash au si multe alte aplicatii atat in criptografie cat si in securitate
- Pentru un fișier x, $H(x)$ poate servi ca identificator unic (amprentă) - existența unui fișier diferit cu același hash implică gasirea de coliziuni in H).
- Prezentam cateva aplicatii care decurg de aici

Alte aplicații ale funcțiilor hash

- *Amprentarea virusilor*
  - scanner-ele de viruși pastreaza hash-urile virusilor cunoscuti
  - verificarea fisierelor potential malitioase se face comparand hash-ul lor cu hash-ul virusilor cunoscuti.
- *Deduplicarea datelor* - stocarea in cloud partajata de mai multi utilizatori - se verifica daca hash-ul unui fisier nou (de incarcat) exista deja in cloud, caz in care se adauga doar un pointer la fisierul respectiv

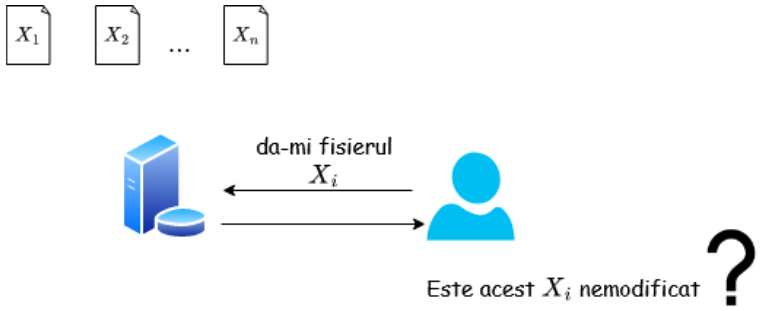
Arbori Merkle

- Un client incarca mai multe fisiere pe server ...



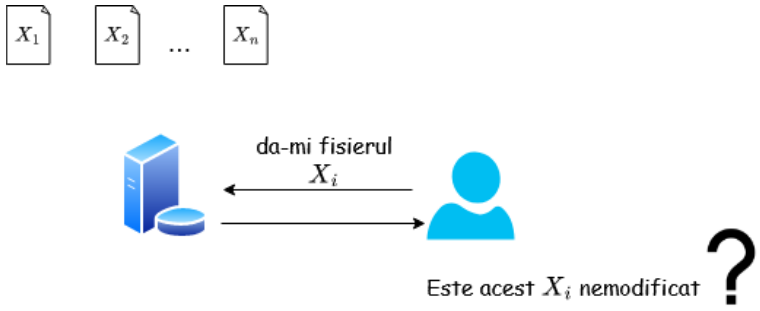
Arbori Merkle

- ... si doreste ca atunci cand le recupereaza de pe server, sa verifice ca nu au fost modificate



1. Prima solutie:
   - clientul stochează local $h_1 = H(x_1), ..., h_n = H(x_n)$ și verifică dacă $h_i = H(x_i')$ pentru fiecare fișier x_i' recuperat
   - Dezavantaj: spațiul necesar stocării crește liniar în n

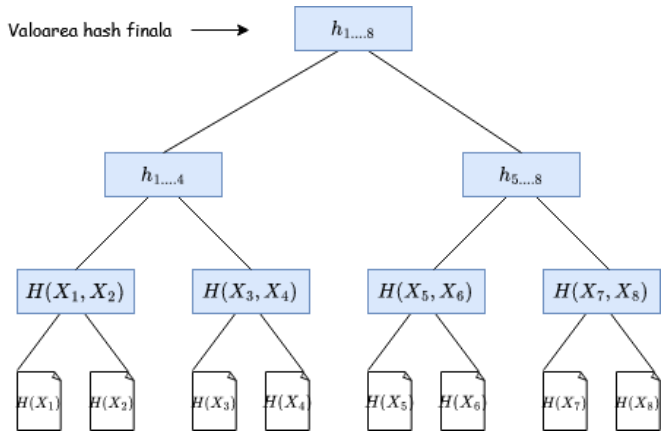
Arbori Merkle



2. A doua soluție
   - clientul stocheaza local un singur hash $h = H(x_1, ..., x_n)$
   - Dezavantaj: pentru a verifica x_i, clientul trebuie să recupereze toate fișierele $x_1, ..., x_n$
   - Soluție: arborii Merkle oferă un compromis între cele două soluții de mai sus

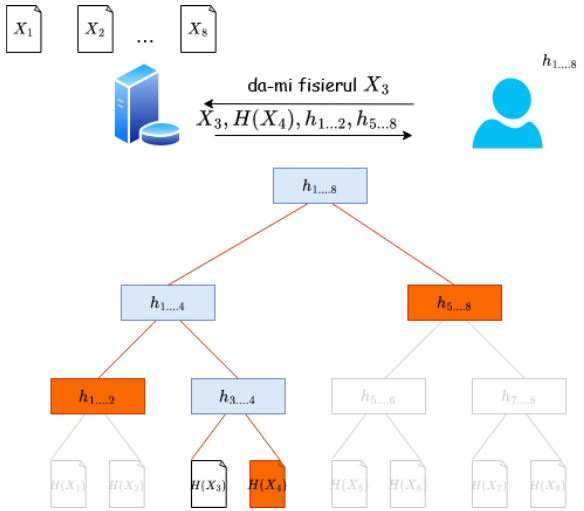
Arbori Merkle

- H - funcție hash rezistentă la coliziuni



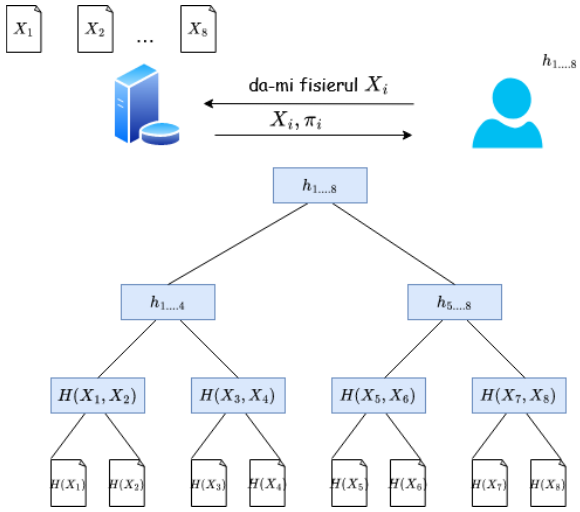
- Este o altenativa la constructia Merkle-Damgard pentru functii hash de lungime arbitrara

Arbori Merkle - stocarea fișierelor în cloud



- user-ul calculează $H(X_3)$, apoi $h'_{3...4} = H(H(X_3), H(X_4))$, $h'_{1...4}$ si $h'_{1...8}$ si verifica daca $h'_{1...8} = h_{1...8}$

Arbori Merkle - stocarea fișierelor în cloud



- π_i - conține nodurile adiacente drumului de la X_i la rădăcină
- pe baza lui π_i, user-ul poate re-calcula valoarea rădăcinii pentru a verifica dacă este aceeași cu valoarea stocată de el

Stocarea parolelor

- Cea mai utilizată metodă de autentificare este bazată pe nume de utilizator și parolă;
- Metoda presupune o etapă de înregistrare, în care utilizatorul alege un *username* și o *parolă* (*pwd*);
- Ulterior, la fiecare logare, utilizatorul introduce cele 2 valori;
- Dacă *username* se regăsește în lista de utilizatori înregistrați și parola introdusă este corectă atunci autentificarea se realizează cu succes;
- În caz contrar, autentificarea eșuază.

Stocarea parolelor

- O greșeală frecventă de implementare o reprezintă afișarea unor mesaje de tipul:

  Nume de utilizator inexistent
  Parolă greșită!

- Întrebare: De ce nu este indicată utilizarea unor astfel de mesaje de eroare?
- Răspuns: Pentru că oferă informații suplimentare adversarului!
- Corect este să se întoarcă un mesaj de eroare generic de tipul:

  Nume de utilizator sau parolă incorecte!

Stocarea parolelor

- O greșeală majoră este stocarea parolelor în clar!
- Întrebare: De ce parolele NU trebuie stocate în clar?
- Răspuns: Pentru că dacă adversarul capătă acces la fisierul de parole atunci află direct parolele tuturor utilizatorilor!
- Pentru stocarea parolelor se utilizează funcțiile hash;
- În fișierul de parole (sau baza de date) se stochează, pentru fiecare utilizator perechi de forma:

  (username, H(pwd))

Stocarea parolelor

- Pentru autentificare, utilizatorul introduce numele de utilizator *username* și parola *pwd*;
- Sistemul de autentificare calculează $H(pwd)$ și se verifică dacă valoarea obținută este stocată în fișierul de parole pentru utilizatorul indicat prin *username*;
- Dacă da, atunci autentificarea se realizează cu succes; în caz contrar, autentificarea eșuază;
- Funcțiile hash sunt funcții **one-way**: cunoscând $H(pwd)$ nu se poate determina *pwd*;
- Această metodă de stocare a parolelor introduce deci avantajul că nu oferă adversarului acces direct la parole, chiar dacă acesta deține fișierul de parole.

Atac de tip dicționar

- Adversarul poate însă să verifice, pe rând, toate parolele cu probabilitate mare de apariție;
- Acestea sunt de obicei cuvinte cu sens sau parole uzuale;
- Se consideră că formează termenii unui dicționar, de unde și numele atacului: **atac de tip dicționar**;
- Pentru a minimiza șansele unor astfel de atacuri:
  - se blochează procesul de autentificare după un anumit număr de încercări nereușite;
  - se obligă utilizatorul să folosească o parolă care satisface anumite criterii : o lungime minimă, utilizarea a cel puțin 3 tipuri de simboluri (litere mici, litere mari, cifre, caractere speciale);
- În caz de succes, adversarul determină parola unui singur utilizator;

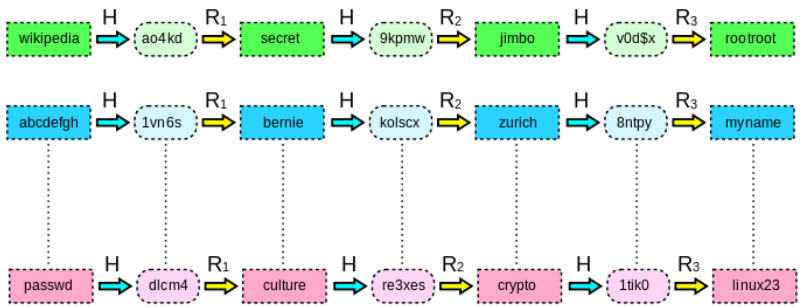
Atac folosind tabele hash precalculate

- Pentru a determina parolele mai multor utilizatori simultan, un adversar poate **precalcula** valorile hash ale parolelor din dicționar;
- Dacă adversarul capătă acces la fișierul de parole, atunci verifică valorile care se regăsesc în lista precalculată;
- Toate conturile utilizatorilor pentru care se potrivesc valorile sunt compromise;
- Atacul necesită capacitate de stocare mare: trebuie stocate toate perechile $(pwd, H(pwd))$ unde pwd este o parolă din dicționar;

Rainbow tables

- **Rainbow tables** introduc un compromis între capacitatea de stocare și timp;
- Se compun lanțuri de lungime t:
  $pwd_1 \rightarrow^H H(pwd_1) \rightarrow^f pwd_2 \rightarrow^H H(pwd_2) \rightarrow^f \cdots \rightarrow^f pwd_{t-1} \rightarrow^H H(pwd_{t-1}) \rightarrow^f pwd_t \rightarrow^H H(pwd_t)$
- f este o funcție de mapare a valorilor hash în parole;
- Atenție! Funcția f nu este inversa funcției H (s-ar pierde proprietatea de *unidirecționalitate* a funcției hash)
- Se memorează doar capetele lanțurilor, valorile intermediare se generează la nevoie;
- Dacă t este suficient de mare, atunci capacitatea de stocare scade semnificativ;

Rainbow tables



[Wikipedia]

Rainbow tables

- Algoritmul de determinare a unei parole:
  1. Se caută valoarea hash a parolei în lista de valori hash a tabelei stocate;
     - 1.1 Dacă se găsește, atunci lanțul căutat este acesta și se trece la pasul 2;
     - 1.2 Dacă nu se găsește, se reduce valoarea hash prin funcția de reducere f într-o parolă căreia i se aplică funcția H și se reia cautarea de la pasul 1;
  2. Se generează întreg lanțul plecând de la valoarea inițială stocată. Parola corespunzătoare este cea situată în lanț înainte de valoarea hash căutată.

Salting

- Pentru a evita atacurile precedente, se utilizează tehnica de salting;
- La înregistrare, se stocheaza pentru fiecare utilizator:
$$(username, salt, H(pwd||salt))$$
- *salt* este o secvență aleatoare de n biți, distinctă pentru fiecare utilizator;
- Adversarul nu poate precalcula valorile hash înainte de a obține acces la fișierul de parole...
- ... decât dacă folosește 2^n valori posibile *salt* pentru fiecare parolă;
- Atacurile devin deci impracticabile pentru n suficient de mare.

Salting

- În plus, în practică se folosesc funcții hash lente;
- Astfel verificarea unui număr mare de parole devine impracticabilă în timp real;
- Un alt avantaj introdus de tehnica de salting este că deși 2 utilizatori folosesc aceeași parolă, valorile stocate sunt diferite:
$$(Alice, 1652674, H(parolatest||1652674))$$
$$(Bob, 3154083, H(parolatest||3154083))$$
- Prin simpla citire a fișierului de parole, adversarul nu își poate da seama că 2 utilizatori folosesc aceeași parolă.

Parole de unică folosință

- Cazul extrem de schimbare periodică a parolei îl reprezintă parolele de unică folosință;
- Utilizatorul folosește o listă de parole, la fiecare logare utilizând următoarea parolă din listă;
- Această lisă se calculează pornind de la o valoare x folosind o funcție hash H;
- Să considerăm o listă de 3 parole de unică folosință:
$$P_0 = H(H(H(H(x))))$$
$$P_1 = H(H(H(x)))$$
$$P_2 = H(H(x))$$
$$P_3 = H(x)$$

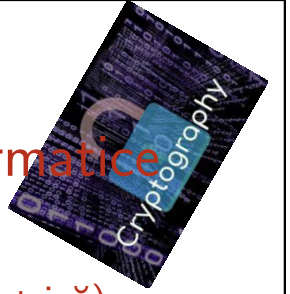
Parole de unică folosință

- La înregistrare, în fișierul de parole se păstrează tripletul:
$$(username, 1, P_0 = H(H(H(H(x)))))$$
- Utilizarorul introduce o parolă P_1 și autentificarea se realizează cu succes dacă $H(P_1) = P_0$;
- Se actualizează fișierul de parole cu noua parolă introdusă:
$$(username, 2, P_1 = H(H(H(x))))$$
- Procesul continuă până se ajunge la $H(x)$;
- Fiind cunoscută o parolă din secvență, se poate calcula imediat parola anterioară, dar NU se poate calcula parola următoare.

Important de reținut!

- Nu păstrați parolele în clar!
- Utilizați mecanisme de tip salting!

Securitatea Sistemelor Informatice



- Curs 8.3 -
Criptografie cu cheie publică (asimetrică)

Adela Georgescu

Facultatea de Matematică și Informatică
Universitatea din București
Anul universitar 2022-2023, semestrul I

Limitările criptografiei simetrice

- Am studiat până acum criptografia simetrică;
- Aceasta asigură confidențialitatea și integritatea mesajelor transmise pe canale nesecurizate;
- Însă rămân multe probleme nerezolvate...

Problema 1 - Distribuirea cheilor

- Criptarea simetrică necesită o cheie secretă comună părților comunicante;
- Întrebare: Cum se obțin și se distribuie aceste chei?
- **Varianta 1**. Se transmit printr-un canal de comunicație nesecurizat;
- NU! Un adversar pasiv le poate intercepta și utiliza ulterior pentru decriptarea comunicației.

Problema 1 - Distribuirea cheilor

- **Varianta 2**. Se transmit printr-un canal de comunicație sigur care presupune un serviciu de mesagerie de încredere;
- Opțiune posibilă la nivel guvernamental sau militar;
- Dar nu va fi niciodată posibilă în cazul organizațiilor numeroase;
- Presupunem doar cazul în care fiecare manager trebuie să partajeze o cheie secretă cu fiecare subordonat;
- Problemele care apar sunt multiple: pentru fiecare nou angajat este necesară stabilirea cheilor, organizația are sedii în mai multe tări, ...

Problema 2 - Stocarea secretă a cheilor

- Rămânem la exemplul anterior al unei organizații numeroase cu N angajați;
- Întrebare: Câte chei sunt necesare pentru ca fiecare 2 angajați să poată comunica criptat?
- Răspuns: $C_N^2 = N(N-1)/2$;
- La acestea se adaugă cheile necesare pentru accesul la resurse (servere, imprimante, baze de date ...);
- Apare o problemă de **logistică**: foarte multe chei sunt dificil de menținut și utilizat;
- Și apare o problemă de **securitate**: cu cât sunt mai multe chei, cu atât sunt mai dificil de stocat în mod sigur, deci cresc șansele de a fi furate de adversari;

Problema 2 - Stocarea secretă a cheilor

- Sistemele informatice sunt deseori infectate de programe malițioase care fură cheile secrete și le transmit prin internet către atacator;
- Totuși dacă numărul de chei este mic, există soluții de stocare cu securitate crescută;
- Un exemplu îl reprezintă dispozitivele de tip *smartcard*;
- Acestea realizează calculele criptografice folosind cheia stocată, asigurând faptul că niciodată cheia secretă nu ajunge în calculator;
- Capacitatea de stocare a unui smartcard este însă limitată, neputând memora, de exemplu mii de chei criptografice.

Problema 3 - Medii de comunicare deschise

- Deși dificil de stocat sau utilizat, criptografia simetrică ar putea (cel puțin în teorie) să rezolve aceste probleme;
- Dar este insuficientă în medii deschise, în care participanții nu se întâlnesc niciodată;
- Astfel de exemple includ: o tranzație prin internet sau un e-mail transmis unei persoane necunoscute;

"Solutions that are based on private-key cryptography are not sufficient to deal with the problem of secure communication in open systems where parties connot phisically meet, or where parties have transient interactions."

(J.Katz, Y.Lindell: Introduction to Modern Cryptography)

Problema 4 - Imposibilitatea non-repudierii

- O cheie simetrică este deținută de cel puțin 2 părți;
- Este imposibil de demonstrat de exemplu că un MAC a fost produs de una dintre cele 2 părți comunicante;
- De aceea nu se poate utiliza autentificarea simetrică pentru a atesta sursa unui mesaj sau document.

Criptografia asimetrică

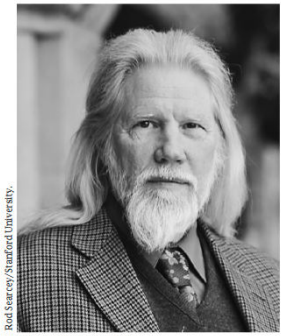
- Criptografia cu cheie publică este introdusă de W.Diffie și M.Hellman in 1976 ca o soluție la problemele enumerate anterior:

"Two kinds of contemporary developments in cryptography are examined. Widening applications of teleprocessing have given rise to a need for new types of cryptographic systems, which minimize the need for secure key distribution channels and supply the equivalent of written signature. This paper suggests ways to solve these currently open problems."

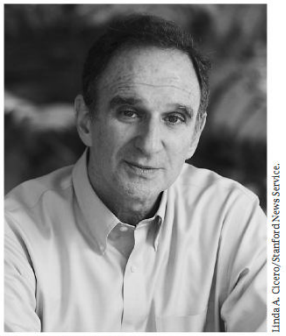
(W.Diffie, M.Hellman: New Directions in Cryptography - abstract)

Criptografia asimetrică

Diffie and Hellman Receive 2015 Turing Award



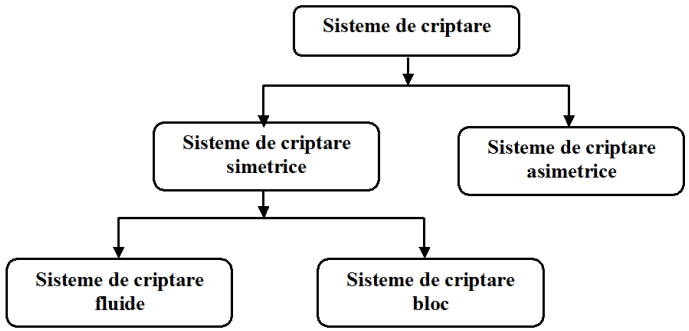
Whitfield Diffie



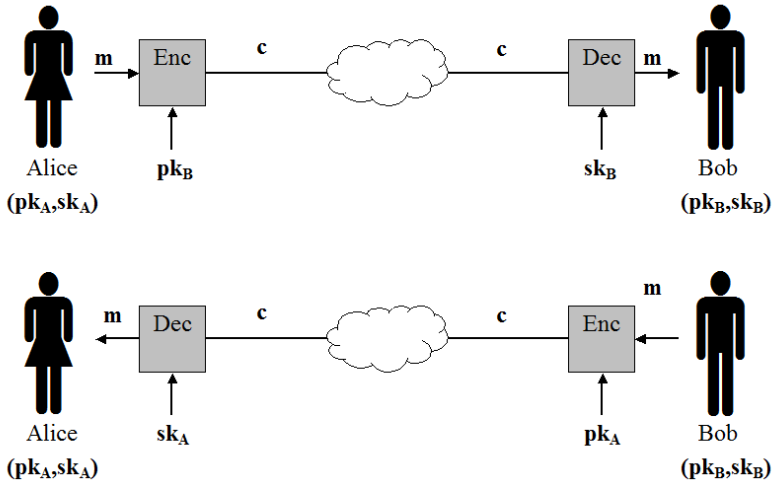
Martin E. Hellman

Sisteme de criptare asimetrice

- Am studiat sisteme de criptare care folosesc **aceeași cheie** pentru criptare și decriptare - sisteme de criptare simetrice;
- Vom studia sisteme de criptare care folosesc **chei diferite** pentru criptare și decriptare - sisteme de criptare asimetrice;



Criptarea asimetrică (cu cheie publică)



Criptarea asimetrică (cu cheie publică)

Definiție

Un sistem de criptare asimetric definit peste $(\mathcal{M}, \mathcal{C})$, cu:

- $\mathcal{M}$ = *spațiul textelor clare (mesaje)*
- $\mathcal{C}$ = *spațiul textelor criptate*

este un triplet $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$*, unde:*

1. $\mathrm{Gen}(1^n)$ - *generează cheile* (pk, sk)
2. Enc - *primește la intrare o cheie publică pk și un mesaj m și calculează* $c \leftarrow \mathrm{Enc}_p k(m)$
3. Dec - *primește la intrare o cheie secretă pk și un mesaj criptat c și intoarce mesajul clar sau eroare (simbolul* $\perp$*)*

a.î. $\forall m \in \mathcal{M}, (pk, sk)$ *generate cu algoritmul* $Gen(1^n)$ $Dec_{sk}(Enc_{pk}(m)) = m$.

Terminologie

- Spre deosebire de criptarea simetrică, criptarea asimetrică folosește o pereche de chei:
- Cheia publică **pk** este folosită pentru criptare;
- Cheia secretă **sk** este folosită pentru decriptare;
- Cheia publică este larg răspândită pentru a asigura posibilitatea de criptare oricui dorește să transmită un mesaj către entitatea căreia îi corespunde;
- Cheia secretă este privată și nu se cunoaște decât de entitatea căreia îi corespunde;
- Considerăm (pentru simplitate) că ambele chei au lungime cel puțin n biți.

Criptografia asimetrică vs. Criptografia simetrică

Criptografia simetrică

- necesită secretizarea întregii chei
- folosește aceeași cheie pentru criptare și decriptare
- rolurile emițătorului și ale receptorului pot fi schimbate
- pentru ca un utilizator să primească mesaje criptate de la mai mulți emițători, trebuie să partajeze cu fiecare câte o cheie

Criptografia asimetrică

- necesită secretizarea unei jumătăți din cheie
- folosește chei distincte pentru criptare și decriptare
- rolurile emițătorului și ale receptorului nu pot fi schimbate
- o pereche de chei asimetrice permite oricui să transmită informație criptată către entitatea căreia îi corespunde

Criptografia asimetrică

Avantaje

- număr mai mic de chei
- simplifică problema distribuirii cheilor
- fiecare participant trebuie să stocheze o singură cheie secretă de lungă durată
- permite comunicarea sigură pe canale publice
- rezolvă problema mediilor de comunicare deschise

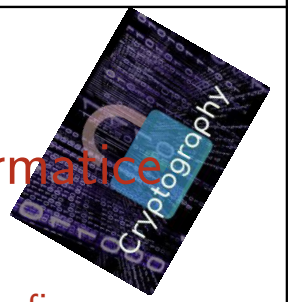
Dezavantaje

- criptarea asimetrică este mult mai lentă decât criptarea simetrică
- compromiterea cheii private conduce la compromiterea tuturor mesajelor criptate primite, indiferent de sursă
- necesită verificarea autenticității cheii publice (PKI rezolvă această problemă)

Important de reținut!

- Criptografia simetrică NU rezolvă toate problemele criptografiei
- Criptografia asimetrică apare în completarea criptografiei simetrice

Securitatea Sistemelor Informatice



- Curs 8.4 -
Teoria numerelor pentru criptografie

Adela Georgescu

Facultatea de Matematică și Informatică
Universitatea din București
Anul universitar 2022-2023, semestrul I

Notații

- $\mathbb{Z} = \{..., -2, -1, 0, 1, 2, ...\}$
- $\mathbb{N} = \{0, 1, 2, ...\}$
- $\mathbb{Z}_+ = \{..., -2, -1, 0, 1, 2, ...\}$
- Pentru $a, n \in \mathbb{N}$ notăm $gcd(a, n)$ ca fiind cel mai mare divizor comun (greatest commun divisor) al lui a și n.
- Exemplu: $gcd(30, 50) = 10$.

Intregi modulo N

- pentru $n \in \mathbb{Z}_+$ notăm
  - $\mathbb{Z}_n = \{0, 1, ..., n-1\}$
  - $\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n \mid gcd(a, n) = 1\}$
  - $\phi(n) = |\mathbb{Z}_n^*|$
- Exemplu: n=12
  - $\mathbb{Z}_{12} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$
  - $\mathbb{Z}_{12}^*$ ={1,5,7,11}
  - $\phi(12) = 4$

Impărțire și rest

- $a = qn + r$ cu $0 \leq r < n$.
  Considerăm împărțirea lui a la n.
  Atunci q este catul împărțirii iar r este restul și notăm

$$a \bmod n = r$$

- Exemplu: $17 \bmod 3 = 2$.
- $a = b \bmod n$ dacă $a \bmod n = b \bmod n$.

Grupuri și invers

- Dacă $n \in \mathbb{Z}_+$ atunci $G = \mathbb{Z}_n^*$ împreună cu operația "*" definită

$$a * b = ab \bmod n$$

pentru $a, b \in G$ formează un grup si are cele trei proprietăți:

- asociativitatea: operatia * este asociativă
- element neutru: există un element $1 \in G$ așa încat $a * 1 \bmod n = 1 * a \bmod n = a$, $\forall a \in G$.
- element inversabil: pentru orice $a \in G$ exista un unic $b \in G$ asa incat $a * b = b * a = 1 \bmod n$.
  b se numeste inversul lui a și îl notăm cu $a^{-1} \bmod n$.
- Exemplu: $5^{-1} \bmod 12$ este acel număr $b \in G$ care satisface $5b \bmod 12 = 1$
- deci b = 5.

Scurtături computaționale

- calculați $5 * 8 * 10 * 16 \bmod 21$.
- Prima variantă: Calculăm mai întai $5 * 8 * 10 * 16 = 6400$ si apoi calculam $6400 \bmod 21 = 16$
- A doua variantă (mai rapidă):
  - $5 * 8 \bmod 21 = 40 \bmod 21 = 19$
  - $19 * 10 \bmod 21 = 190 \bmod 21 = 1$
  - $1 * 16 \bmod 21 = 16$

Ordinul unui grup

- Ordinul unui grup G este numărul de elemente din acel grup, îl notăm cu $|G|$.
- Exemplu: Ordinul lui $\mathbb{Z}_{21}^* = 12$ pentru că

$$\mathbb{Z}_{21}^* = \{1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20\}$$

- Fie G un grup de ordin m si $a \in G$. Atunci:
  - $a^m = 1$.
  - Pentru orice $i \in \mathbb{Z}$, $a^i = a^{i \bmod m}$.
  - Exemplu: Calculați 5^{74} mod 21.
  - Răspuns: Fie $\mathbb{Z}_{21}^*$ si $a = 5$. Atunci $m = 12$ și
  - $5^{74} \bmod 21 = 5^{74 \bmod 12} \bmod 21 = 5^2 \bmod 21 = 4$.

Securitatea Sistemelor Informatice

- Curs 8.5 -
Prezumpții criptografice dificile

Adela Georgescu

Facultatea de Matematică și Informatică
Universitatea din București
Anul universitar 2022-2023, semestrul I

Prezumpții criptografice dificile

- Criptografia modernă se bazează pe prezumpția că *anumite* probleme nu pot fi rezolvate în timp polinomial;
- Până acum am văzut că schemele de criptare și de autentificare se bazează pe prezumpția existenței permutărilor pseudoaleatoare;
- Dar această prezumpție e nenaturală și foarte puternică;
- În practică, PRF pot fi instanțiate cu cifruri bloc;
- Însă metode pentru a demonstra pseudoaleatorismul construcțiilor practice relativ la alte prezumpții "mai rezonabile" nu se cunosc;
- Dar e posibil a demonstra existența permutărilor pseudoaleatoare pe baza unei prezumpții mult mai slabe, cea a existenței funcțiilor one-way;

Prezumpții criptografice dificile

- În continuare vom introduce câteva două considerate "dificile": problema factorizării și problema logaritmului discret și vom prezenta funcții conjecturate ca fiind one-way bazate pe aceste probleme;
- Tot materialul ce urmează se bazează pe noțiuni de teoria numerelor;
- La criptografia *simetrică* (cu cheie secretă) am văzut primitve criptografice (i.e. funcții hash, PRG, PRF) care pot fi construite eficient fără a implica teoria numerelor;
- La criptografia *asimetrică* (cu cheie publică) construcțiile cunoscute se bazează pe probleme matematice dificile din teoria numerelor;

1. Problema factorizării

- O primă problemă conjecturată ca fiind dificilă este problema factorizării numerelor întregi sau mai simplu problema factorizării;
- Fiind dat un număr compus N, problema cere să se găsească două numere prime x_1 și x_2 pe n biți așa încât $N = x_1 \cdot x_2$;
- Cele mai greu de factorizat sunt numerele care au factori primi foarte mari.

Primalitate și factorizare

"The problem of distinguishing prime numbers from composite numbers and of resolving the latter into their prime factors is known to be one of the most important and useful in arithmetic. It has engaged the industry and wisdom of ancient and modern geometers to such an extent that it would be superfluous to discuss the problem at length... The dignity of the science itself seems to require that every possible means be explored for the solution of a problem so elegant and so celebrated."

(C.F.Gauss 1777 – 1855)

Generarea numerelor prime

- Pentru a putea folosi problema în criptografie, trebuie să generăm numere prime aleatoare *în mod eficient*;
- Putem genera un număr prim aleator pe n biți prin alegerea repetată de numere aleatoare pe n biți până când găsim unul prim;
- Pentru eficiență, ne interesează două aspecte:
  1. probabilitatea ca un număr aleator de n biți să fie prim;
  2. cum testăm eficient că un număr dat p este prim.

Generarea numerelor prime

- Pentru distribuția numerelor prime, se cunoaște următorul rezultat matematic:

Teoremă

Pentru orice $n > 1$, proportia de numere prime pe n biți este de cel puțin $1/3n$.

- Rezultă imediat că dacă testăm $t = 3n^2$ numere, probabilitatea ca un număr prim să nu fie ales este cel mult e^{-n}, deci neglijabilă.
- Deci avem un algoritm în timp polinomial pentru gasirea numerelor prime

Testarea primalității

- Cei mai eficienți algoritmi sunt probabiliști:
  - Dacă numărul p dat este prim atunci algoritmul întotdeauna întoarce rezultatul *prim*;
  - Dacă p este compus, atunci cu probabilitate mare algoritmul va întoarce *compus*;
  - **Concluzie:** dacă outputul este *compus*, atunci p sigur este compus, dacă outputul este *prim*, atunci cu probabilitate mare p este prim dar este posibil și să se fi produs o eroare;
- Un algoritm determinist polinomial a fost propus în 2002, dar este mai lent decât algoritmii probabiliști;
- Un algoritm probabilist foarte răspândit este Miller-Rabin care acceptă la intrare un număr N și rulează în timp polinomial în $|N|$

Algoritmi de factorizare

- Deocamdată nu se cunosc *algoritmi polinomiali* pentru problema factorizării;
- Dar există algoritmi mult mai buni decât forța brută;
- Prezentăm în continuare câțiva algoritmi de factorizare.

Algoritmi de factorizare

- Reamintim: Fiind dat un număr compus N, **problema factorizării** cere să se găsească 2 numere prime p și q a.î. $N = pq$;
- Considerăm $|p| = |q| = n$ și deci $n = O(\log N)$;
- Metoda cea mai simplă este împărțirea numărului N prin toate numerele p impare din intervalul $p = 3, ..., \left\lfloor \sqrt{N} \right\rfloor$.
- Complexitatea timp este $O(\sqrt{N} \cdot (\log N)^c)$ unde c este o constantă, adica exponentiala in numarul b de biti al lui N; $b = \log_2 N$. Complexitatea este $O(N^{1/2}) = O((2^{\log_2 N})^{1/2})$
- Pentru $N < 10^{12}$ metoda este destul de eficientă.

Algoritmi de factorizare

- Există însă algoritmi mai sofisticați, cu timp de execuție mai bun, între care:
  - Metoda **Pollard p − 1**: funcționează atunci când $p - 1$ are factori primi "mici";
  - Metoda **Pollard rho**: timpul de execuție este $O(N^{1/4} \cdot (\log N)^c)$, deci tot exponențial în lungimea lui N;
  - **Algoritmul sitei pătratice** - rulează în timp *sub-exponențial* în lungimea lui N.
- Deocamdată, cel mai rapid algoritm de factorizare este o îmbunătățire a sitei pătratice care factorizează un număr N de lungime $O(n)$ în timp $2^{O(n^{1/3} \cdot (\log n)^{2/3})}$.

RSA Challenge

- În 1991, Laboratoarele RSA lansează *RSA Challenge*;
- Aceasta presupune factorizarea unor numere N, unde $N = p \cdot q$, cu p, q 2 numere prime mari;
- Au fost lansate mai multe provocări, câte 1 pentru fiecare dimensiune (în biți) a lui N:
- Exemple includ: RSA-576, RSA-640, RSA-768, $\cdots$, RSA-1024, RSA-1536, RSA-2048;
- Provocarea s-a încheiat oficial în 2007;
- Multe provocări au fost sparte în cursul anilor (chiar și ulterior închiderii oficiale), însă există numere încă nefactorizate:

RSA Challenge

- **RSA-1024**

135066410865995223349603216278805969938881475605667027524485143851526510604859533833940287150571909441798207282164471551373680419703964191743046496589274256239341020864383202110372958725762358509643110564073501508187510676594629205563685529475213500852879416377328533906109750544334999811150056977236890927563

[http://www.emc.com/emc-plus/rsa-labs/historical/the-rsa-challenge-numbers.htm]

RSA Challenge

- **RSA-2048**

25195908475657893494027183240048398571429282126204032027777137836043662020707595556264018525880784406918290641249515082189298559149176184502808489120072844992687392807287776735971418347270261896375014971824691165077613379859095700097330459748808428401797429100642458691817195118746121515172654632282216869987549182422433637259085141865462043576798423387184774447920739934236584823824281198163815010674810451660377306056201619676256133844143603833904414952634432190114657544454178424020924616515723350778707749817125772467962926386356373289912154831438167899885040445364023527381951378636564391212010397122822120720357

[http://www.emc.com/emc-plus/rsa-labs/historical/the-rsa-challenge-numbers.htm]

2. Problema logaritmului discret

- O altă prezumție dificilă este DLP (Discrete Logarithm Problem) (sau PLD (Problema Logaritmului Discret));
- Considerăm $\mathbb{G}$ un grup ciclic de ordin q;
- Există un generator $g \in \mathbb{G}$ a.î. $\mathbb{G} = \{g^0, g^1, ..., g^{q-1}\}$;
- Echivalent, pentru fiecare $h \in \mathbb{G}$ există un *unic* $x \in \mathbb{Z}_q$ a.î. $g^x = h$;
- x se numește logaritmul discret al lui h în raport cu g și se notează

$$x = \log_g h$$

Experimentul logaritmului discret $DLog_{\mathcal{A}}(n)$

1. Generează $(\mathbb{G}, q, g)$ unde $\mathbb{G}$ este un grup ciclic de ordin q (cu $|q| = n$) iar g este un generator al lui $\mathbb{G}$.
2. Alege $h \leftarrow^R \mathbb{G}$. (se poate alege $x' \leftarrow^R \mathbb{Z}_q$ și apoi $h := g^{x'}$.)
3. $\mathcal{A}$ primește $\mathbb{G}, q, g, h$ și întoarce $x \in \mathbb{Z}_q$;
4. Output-ul experimentului este 1 dacă $g^x = h$ și 0 altfel.

Definiție

Spunem că problema logaritmului discret (DLP) este dificilă dacă pentru orice algoritm PPT $\mathcal{A}$ există o funcție neglijabilă negl *așa încât*

$$Pr[DLog_{\mathcal{A}}(n) = 1] \leq negl(n)$$

Grupuri ciclice de ordin prim

- Există câteva clase de grupuri ciclice pentru care DLP este considerată dificilă;
- Una dintre ele este clasa grupurilor ciclice de *ordin prim* (în aceste grupuri, problema este "cea mai dificilă");
- DLP nu poate fi rezolvată în timp polinomial în grupurile care nu sunt de ordin prim, ci doar este *mai ușoară*;
- În aceste grupuri căutarea unui generator și verificarea că un număr dat este generator sunt triviale.

Lucrul în $\mathbb{Z}_p^*$

- DLP este considerată dificilă în grupuri ciclice de forma $\mathbb{Z}_p^*$ cu p prim;
- Însă pentru $p > 3$ grupul $\mathbb{Z}_p^*$ NU are ordin prim;
- Aceasta problemă se rezolvă folosind un *subgrup* potrivit al lui $\mathbb{Z}_p^*$;

Important de reținut!

- Cel mai rapid algoritm de factorizare necesită timp sub-exponențial;
- Problema logaritmului discret este dificilă.

Securitatea Sistemelor Informatice

- Curs 9.0 -
Noțiuni de securitate in criptografia asimetrică

Adela Georgescu

Facultatea de Matematică și Informatică
Universitatea din București
Anul universitar 2022-2023, semestrul I

Securitate perfectă

- Începem studiul securității în același mod în care am început la criptografia simetrică: cu securitatea perfectă;
- Definiția e analoagă cu diferența că adversarul cunoaște, în afara textului criptat, și cheia publică;

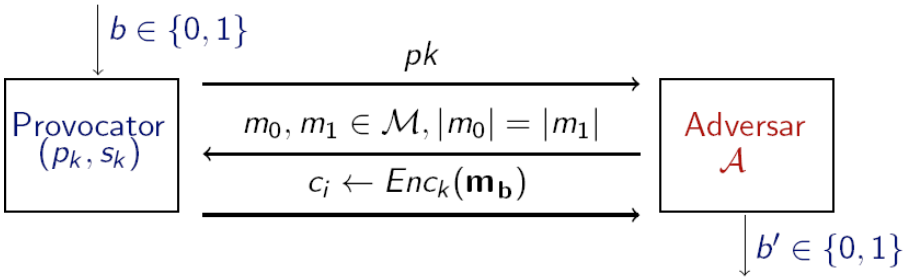
Securitate perfectă

- Întrebare: Securitatea perfectă este posibilă în cadrul criptografiei cu cheie publică?
- Răspuns: NU! Indiferent lungimea cheilor și a mesajelor;
- Având pk și un text criptat $c = Enc_{pk}(m)$, un adversar nelimitat computațional poate determina mesajul m cu probabilitate 1.

Indistinctibilitate

- Indistinctibilitatea în criptografia cu cheie publică este corespondenta noțiunii similare din criptografia cu cheie secretă;
- Vom defini această noțiune pe baza unui experiment de indistinctibilitate $PubK^{eav}_{\mathcal{A},\pi}(n)$ unde $\pi = (Enc, Dec)$ este schema de criptare iar n este parametrul de securitate al schemei π;
- Personaje participante: **adversarul** $\mathcal{A}$ care încearcă să spargă schema și un **challenger**.

Experimentul $PubK^{eav}_{\mathcal{A},\pi}(n)$



- Output-ul experimentului este 1 dacă $b' = b$ și 0 altfel. Dacă $PubK^{eav}_{\mathcal{A},\pi}(n) = 1$, spunem că $\mathcal{A}$ a efectuat experimentul cu succes.

Securitate pentru interceptare simplă

Definiție

O schemă de criptare $\pi = (Enc, Dec)$ este indistinctibilă în prezența unui atacator pasiv dacă pentru orice adversar $\mathcal{A}$ există o funcție neglijabilă negl așa încât

$$Pr[PubK^{eav}_{\mathcal{A},\pi}(n) = 1] \leq \frac{1}{2} + negl(n).$$

Securitate pentru interceptare simplă

- Principala diferență față de definiția similară studiată la criptografia cu cheie secretă este că $\mathcal{A}$ primește cheia publică pk;
- Adică $\mathcal{A}$ primește acces *gratuit* la un oracol de criptare, ceea ce înseamnă că el poate calcula $Enc_{pk}(m)$ pentru orice m;
- Prin urmare, definiția este echivalentă cu cea pentru securitate CPA (nu mai este necesar oracolul de criptare pentru că $\mathcal{A}$ își poate cripta singur mesajele);
- Reamintim că în criptografia simetrică există scheme indistinctibil sigure dar care nu sunt CPA-sigure .

Insecuritatea schemelor deterministe

- După cum am văzut la criptografia simetrică, nici o schemă deterministă nu poate fi CPA sigură;
- Datorită echivalenței între noțiunile de securitate CPA și indistinctibilitate pentru interceptare simplă (în criptografia asimetrică) concluzionăm că:

Teoremă

Nici o schemă de criptare cu cheie publică deterministă nu poate fi semantic sigură pentru interceptarea simplă.

Securitate CCA

- Noțiunea de securitate CCA rămâne identică cu cea de la sistemele simetrice;
- Capabilitățile adversarului: el poate interacționa cu un **oracol de decriptare**, fiind un adversar *activ* care poate rula atacuri în timp polinomial;
- Adversarul poate transmite către oracolul de decriptare *anumite* mesaje c și primește înapoi mesajul clar corespunzător;
- Ca și în cazul securității CPA, adversarul nu mai necesită acces la oracolul de criptare pentru că deține cheia publică pk și poate realiza singur criptarea oricărui mesaj m.

Important de reținut!

- În criptografia cu cheie publică:
  - NU există securitate perfectă
  - indistinctibilitate = securitate CPA

Securitatea Sistemelor Informatice

- Curs 9.1 -
Criptare hibridă

Adela Georgescu

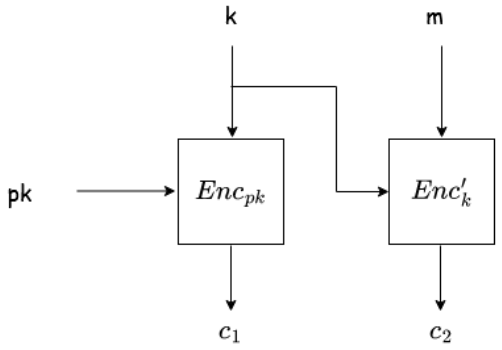
Facultatea de Matematică și Informatică
Universitatea din București
Anul universitar 2022-2023, semestrul I

Criptarea hibridă

- Criptarea cu cheie secretă este mult mai rapidă decat criptarea cu cheie publică
- Pentru mesajele care sunt suficient de lungi, se folosește criptare cu cheie secretă în tandem cu criptarea cu cheie publică;

Criptarea hibridă

- Rezultatul acestei combinații se numește criptare hibridă și este folosită extensiv în practică;



Criptare hibridă

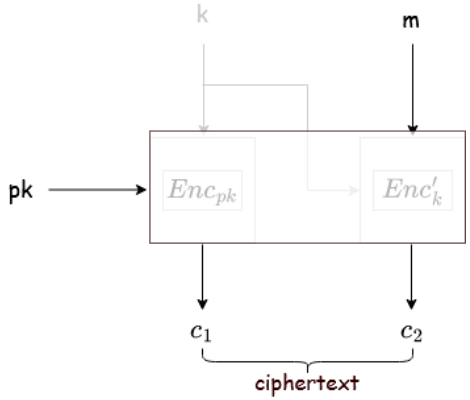
- Pentru criptarea unui mesaj m, se urmează doi pași:

1. Expeditorul alege aleator o cheie k pe care o criptează folosind cheia publică a destinatarului, rezultând $c_1 = Enc_{pk}(k)$; Numai destinatarul va putea decripta k, ea rămânând secretă pentru un adversar;
2. Expeditorul criptează m folosind o schemă de criptare cu cheie secretă (Enc', Dec') cu cheia k, rezultând $c_2 = Enc'_k(m)$;

- Mesajul criptat este $c = (c_1, c_2)$;

Criptare hibridă

- Construcția este o schemă de criptare asimetrică (cele două părți nu partajează o cheie secretă în avans).



Securitate

Teoremă

Dacă Π *este o schemă de criptare cu cheie publică CPA-sigură iar* Π' *este o schemă de criptare cu cheie secretă sigură semantic, atunci construcția hibridă* Π^{hyb} *este o schemă de criptare cu cheie publică CPA-sigură.*

- Este suficient ca Π' să satisfacă noțiunea mai slabă de securitate semantică (care nu implică securitate CPA)...
- ...deoarece cheia secretă k este una "nouă" și aleasă aleator de fiecare dată când se criptează un mesaj;
- Cum o cheie k este folosită o singură dată, e suficientă noțiunea de securitate la interceptare simplă pentru securitatea schemei hibride.

Important de reținut!

- Pentru criptarea mesajelor lungi, în practică se folosește criptarea hibridă
- Aceasta îmbină avantajele criptării simetrice și criptării asimetrice

Securitatea Sistemelor Informatice

- Curs 9.2 -
RSA

Adela Georgescu

Facultatea de Matematică și Informatică
Universitatea din București
Anul universitar 2022-2023, semestrul I

Funcții one-way

- Reprezinta o primitiva criptografica minima, necesara si suficienta pentru criptarea cu cheie secreta dar si pentru codurile de autentificare a mesajelor
- O functie f one-way este usor de calculat si dificil de inversat

Definiție

O funcție $f : \{0,1\}^* \to \{0,1\}^*$ este one-way dacă următoarele două condiții sunt îndeplinite:

1. $\mathbb{U}$*șor de calculat: Exista un algoritm polinomial pentru calculul lui f*
2. $\mathbb{D}$*ificil de inversat: Pentru orice algoritm polinomial $\mathcal{A}$, exista o functie neglijabila negl asa incat*

$$Pr[Invert_{\mathcal{A},f}(n) = 1] \leq negl(n)$$

Functii one-way

- Problema factorizarii numerelor mari in produs de doua numere prime de aceeasi lungime este one-way ...
- ... însa nu poate fi folosita direct pentru criptografie
- In schimb, introducem o problema apropiata de problema factorizarii pe baza careia putem construi sisteme de criptare

Problema RSA

- Problema RSA se bazează pe dificultatea factorizării numerelor mari: $N = p \cdot q$, p și q prime;
- Fie $\mathbb{Z}_N^*$ un grup de ordin $\phi(N) = (p-1)(q-1)$;
- Dacă se cunoaște factorizarea lui N, atunci $\phi(N)$ este ușor de calculat
- Fixăm e cu $gcd(e, \phi(N)) = 1$. Atunci $(x^e)^d = x^{ed \bmod \phi(N)} = x \bmod N = \bmod N = (x^d)^e$
- x^d se numeste radacina de ordin e a lui x modulo N
- Dacă p și q se cunosc, atunci putem calcula $\phi(N)$ și $d = e^{-1} \bmod \phi(N)$
- Daca p și q nu se cunosc
  - calculul lui $\phi(N)$ este la fel de dificil precum factorizarea lui N
  - calculul lui d este la fel de dificil precum factorizarea lui N

Experimentul RSA $RSA - inv_{\mathcal{A},GenRSA(n)}$

- Consideram algoritmul $\mathrm{GenRSA}(1^n)$ care are ca output (N, e, d) unde $ed = 1 \bmod \phi(N)$
- Considerăm experimentul RSA pentru un algoritm $\mathcal{A}$ și un parametru n.
  1. Execută GenRSA și obține (N, e, d);
  2. Alege $y \leftarrow \mathbb{Z}_N^*$;
  3. $\mathcal{A}$ primește N, e, y și întoarce $x \in \mathbb{Z}_N^*$;
  4. Output-ul experimentului este 1 dacă $x^e = y \bmod N$ și 0 altfel.

Definiție

Spunem că problema RSA este dificilă cu privire la GenRSA dacă pentru orice algoritm PPT $\mathcal{A}$ există o funcție neglijabilă negl *așa încât*

$$Pr[RSA - inv_{\mathcal{A},GenRSA(n)} = 1] \leq negl(n)$$

GenRSA

- Prezumpția RSA este că există un algoritm GenRSA pentru care problema RSA este dificilă;
- Un algoritm GenRSA poate fi construit pe baza unui număr compus împreună cu factorizarea lui;

Algorithm 1 GenRSA

Input: n

Output: N, e, d

1: **genereaza** p și q prime pe n-biti; $N = p \cdot q$
2: $\phi(N) = (p-1)(q-1)$
3: **găsește** e a.î. $gcd(e, \phi(N)) = 1$
4: **calculează** $d := e^{-1} \bmod \phi(N)$
5: **return** N, e, d

GenRSA

- Valoarea lui e aleasa pare ca nu afecteaza dificultatea problemei RSA
- în practică se folosește $e = 3$ sau $e = 16$ pentru exponențiere eficientă
- Daca N este usor de factorizat, atunci problema RSA este usoara
- Pentru ca problema RSA să poată fie dificilă, trebuie ca N-ul ales în GenRSA să fie dificil de factorizat în produs de două numere prime;
- Nu se cunoaște nici o dovadă că nu există o altă metodă de a rezolva problema RSA care să nu implice calculul lui $\phi(N)$ sau al lui d.

Exemplu

- Presupunem $(N, p, q) = (143, 11, 13)$. Atunci $\phi(N) = 120$.
- Alegem e asa incat $gcd(e, \phi(N)) = 1$, fie $e = 7$.
- Calculăm $d = e^{-1} \bmod \phi(N)$ si obtinem $d = 103$. Deci output-ul algoritmului GenRSA este (143,7,103).

Textbook RSA

- Definim sistemul de criptare *Textbook RSA* pe baza problemei prezentată anterior;
  1. Se rulează GenRSA pentru a determina N, e, d.
     - Cheia publică este: $pk = (N, e)$;
     - Cheia privată este $sk = d$;
  2. **Enc**: dată o cheie publică (N, e) și un mesaj $m \in \mathbb{Z}_N$, întoarce $c = m^e \mod N$;
  3. **Dec**: dată o cheie secretă (N, d) și un mesaj criptat $c \in \mathbb{Z}_N$, întoarce $m = c^d \mod N$.
- Sistemul de criptare este corect pentru ca $\mathbf{Dec}_{sk}(\mathbf{Enc}_{pk}(m)) = m$ astfel:
  $(m^e)^d \bmod N = m^{ed \bmod \Phi(N)} \bmod N = m^1 \bmod N = m$

Securitate - Problema 1

Problema 1: **Determinismul**

- **Întrebare**: Este Textbook RSA CPA-sigur sau CCA-sigur?
- **Răspuns**: NU! Sistemul este determinist, deci nu poate rezista definițiilor de securitate!

Securitate - Problema 2

Problema 2: **Utilizarea multiplă a modulului**

- Cunoscând e, d, N cu $(e, \phi(N)) = 1$ se poate determina eficient factorizarea lui N;
- **Întrebare**: Este corect să se utilizeze mai multe perechi de chei care folosesc același modul?
- **Răspuns**: NU! Fie 2 perechi de chei:

$$pk_1 = (N, e_1);\ sk_1 = (N, d_1)$$
$$pk_2 = (N, e_2);\ sk_2 = (N, d_2)$$

- Posesorul perechii (pk_1, sk_1) factorizează N, apoi determină $d_2 = e_2^{-1} \mod \phi(N)$.

Important de reținut!

- RSA este cel mai cunoscut și mai utilizat algoritm cu cheie publică;
- Textbook RSA NU trebuie utilizat!

Securitatea Sistemelor Informatice

- Curs 9.3 -
PKCS

Adela Georgescu

Facultatea de Matematică și Informatică
Universitatea din București
Anul universitar 2022-2023, semestrul I

Padded RSA

- Am văzut că Textbook RSA este nesigur;
- Eliminăm una dintre problemele principale, aceea este că sistemul este determinist;
- Introducem în acest sens Padded RSA;
- Ideea este să se adauge un număr aleator (*pad*) la mesajul clar înainte de criptare;
- Notăm în continuare cu n parametrul de securitate (conform GenRSA).

Padded RSA

1. Se rulează GenRSA pentru a determina N, e, d.
   - Cheia publică este: (N, e);
   - Cheia privată este (N, d);
2. **Enc**: dată o cheie publică (N, e) și un mesaj $m \in \{0,1\}^{l(n)}$, alege $r \leftarrow^R \{0,1\}^{|N|-l(n)-1}$, interpretează $r||m$ ca un element în $\mathbb{Z}_N$ și întoarce $c = (r||m)^e \mod N$;
3. **Dec**: dată o cheie secretă (N, d) și un mesaj criptat $c \in \mathbb{Z}_N$, calculează $c^d \mod N$ și întoarce ultimii $l(n)$ biți.

Padded RSA

- Pentru $l(n)$ foarte mare, atunci este posibil un atac prin forță brută care verifică toate valorile posibile pentru r;
- Pentru $l(n)$ mic se obține securitate CPA:

Teoremă

Dacă problema RSA este dificilă, atunci Padded RSA cu $l(n) = O(\log n)$ este CPA-sigură.

PKCS #1 v1.5

- martie 1998 - Laboratoarele RSA introduc PKCS #1 v1.5;
- PKCS = Public-Key Cryptography Standard;
- Folosește Padded RSA;
- Utilizat în HTTPS, SSL/TLS, XML Encryption.

PKCS #1 v1.5

- Notăm k lungimea modulului N în bytes: $2^{8(k-1)} \leq N < 2^{8k}$;
- Mesajele m care se criptează se consideră multiplii de 8 biți, de maxim $k - 11$ bytes;
- Criptarea se realizează astfel:

$$(00000000||00000010||r||00000000||m)^e \mod N$$

- r este ales aleator, pe $k - D - 3$ bytes nenuli, unde D este lungimea lui m în bytes;

Securitatea PKCS #1 v1.5

- Se crede că este CPA-sigur, dar acest lucru nu este demonstrat;
- Cu siguranță însă nu este CCA-sigur;
- În 1998, D.Bleichenbacher publică un atac care bazându-se pe faptul că serverul web (HTTPS) întoarce eroare dacă primii 2 octeți nu sunt **02**;
- Scopul adversarului este să decripteze un text c;
- Adversarul transmite către server $c' = r^e \cdot c \mod N$;
- Răspunsul serverului indică adversarului dacă c' este valid (i.e. începe cu **02**);
- Adversarul folosește răspunsul primit pentru a determina informații despre m;
- Repetă atacul până determină mesajul m.

OAEP

- octombrie 1998 - Laboratoarele RSA introduc un nou standard PKCS demonstrat CCA-sigur...
- ...în modelul $\mathcal{ROM}$ (Random Oracle Model);
- Este vorba despre PKCS #1 v2.0 sau OAEP = Optimal Asymmetric Encryption Standard;

OAEP

- OAEP este de fapt o modalitate de padding;
- OAEP este o metodă **nedeterministă** și **inversabilă** care transformă un mesaj m de lungime $n/2$ într-o secvență m' de lungime $2n$;
- Notăm $m' = OAEP(m, r)$, unde r este o secvență aleatoare (de lungime n);
- RSA-OAEP criptează $m \in \{0,1\}^{n/2}$ folosind cheia publică (N, e) ca:

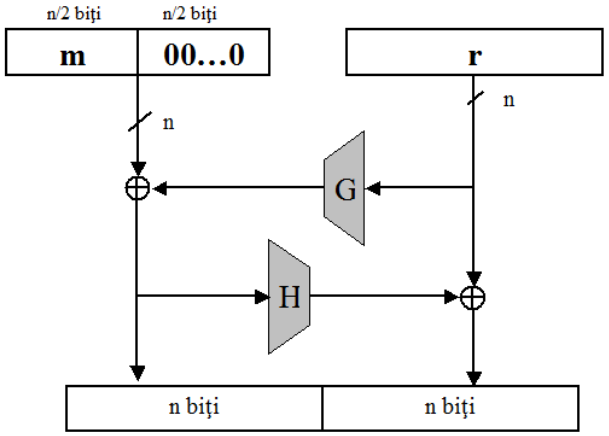
$$(OAEP(m, r))^e \mod N$$

- RSA-OEAP decriptează c folosind cheia secretă (N, d) ca:

$$(m, r) = OAEP^{-1}(c^d \mod N)$$

OAEP

- OAEP este definit astfel:



Notații

- G, H = funcții hash
- $m \in \{0,1\}^{n/2}$ mesajul
- $r \leftarrow^R \{0,1\}^n$
- se obține $OAEP(m, r)$ pe $2n$ biți

Important de reținut!

- Utilizarea RSA în practică:
  PKCS #1 v1.5 și PKCS #1 v2.0 (OAEP)

Securitatea Sistemelor Informatice

- Curs 10.1 -
PKCS

Adela Georgescu

Facultatea de Matematică și Informatică
Universitatea din București
Anul universitar 2022-2023, semestrul I

Textbook RSA

- Definim sistemul de criptare *Textbook RSA* pe baza problemei prezentată anterior;
  1. Se rulează GenRSA pentru a determina N, e, d.
     - Cheia publică este: $pk = (N, e)$;
     - Cheia privată este $sk = d$;
  2. **Enc**: dată o cheie publică (N, e) și un mesaj $m \in \mathbb{Z}_N$, întoarce $c = m^e \mod N$;
  3. **Dec**: dată o cheie secretă (N, d) și un mesaj criptat $c \in \mathbb{Z}_N$, întoarce $m = c^d \mod N$.
- Sistemul de criptare este corect pentru ca $\mathbf{Dec}_{sk}(\mathbf{Enc}_{pk}(m)) = m$ astfel:
  $(m^e)^d \bmod N = m^{ed \bmod \Phi(N)} \bmod N = m^1 \bmod N = m$

Padded RSA

- Am văzut că Textbook RSA este nesigur;
- Eliminăm una dintre problemele principale, aceea este că sistemul este determinist;
- Introducem în acest sens Padded RSA;
- Ideea este să se adauge un număr aleator (*pad*) la mesajul clar înainte de criptare;
- Notăm în continuare cu n parametrul de securitate (conform GenRSA).

Padded RSA

1. Se rulează GenRSA pentru a determina N, e, d.
   - Cheia publică este: (N, e);
   - Cheia privată este (N, d);
2. **Enc**: dată o cheie publică (N, e) și un mesaj $m \in \{0,1\}^{l(n)}$, alege $r \leftarrow^R \{0,1\}^{|N|-l(n)-1}$, interpretează $r||m$ ca un element în $\mathbb{Z}_N$ și întoarce $c = (r||m)^e \mod N$;
3. **Dec**: dată o cheie secretă (N, d) și un mesaj criptat $c \in \mathbb{Z}_N$, calculează $c^d \mod N$ și întoarce ultimii $l(n)$ biți.

Padded RSA

- Pentru $l(n)$ foarte mare, atunci este posibil un atac prin forță brută care verifică toate valorile posibile pentru r;
- Pentru $l(n)$ mic se obține securitate CPA:

Teoremă

Dacă problema RSA este dificilă, atunci Padded RSA cu $l(n) = O(\log n)$ este CPA-sigură.

PKCS #1 v1.5

- martie 1998 - Laboratoarele RSA introduc PKCS #1 v1.5;
- PKCS = Public-Key Cryptography Standard;
- Folosește Padded RSA;
- Utilizat în HTTPS, SSL/TLS, XML Encryption.

PKCS #1 v1.5

- Notăm k lungimea modulului N în bytes: $2^{8(k-1)} \leq N < 2^{8k}$;
- Mesajele m care se criptează se consideră multiplii de 8 biți, de maxim $k - 11$ bytes;
- Criptarea se realizează astfel:

$$(00000000||00000010||r||00000000||m)^e \mod N$$

- r este ales aleator, pe $k - D - 3$ bytes nenuli, unde D este lungimea lui m în bytes;

Securitatea PKCS #1 v1.5

- Se crede că este CPA-sigur, dar acest lucru nu este demonstrat;
- Cu siguranță însă nu este CCA-sigur;
- În 1998, D.Bleichenbacher publică un atac care bazându-se pe faptul că serverul web (HTTPS) întoarce eroare dacă primii 2 octeți nu sunt **02**;
- Scopul adversarului este să decripteze un text c;
- Adversarul transmite către server $c' = r^e \cdot c \mod N$;
- Răspunsul serverului indică adversarului dacă c' este valid (i.e. începe cu **02**);
- Adversarul folosește răspunsul primit pentru a determina informații despre m;
- Repetă atacul până determină mesajul m.

OAEP

- octombrie 1998 - Laboratoarele RSA introduc un nou standard PKCS demonstrat CCA-sigur...
- ...în modelul $\mathcal{ROM}$ (Random Oracle Model);
- Este vorba despre PKCS #1 v2.0 sau OAEP = Optimal Asymmetric Encryption Standard;

OAEP

- OAEP este de fapt o modalitate de padding;
- OAEP este o metodă **nedeterministă** și **inversabilă** care transformă un mesaj m de lungime $n/2$ într-o secvență m' de lungime $2n$;
- Notăm $m' = OAEP(m, r)$, unde r este o secvență aleatoare (de lungime n);
- RSA-OAEP criptează $m \in \{0,1\}^{n/2}$ folosind cheia publică (N, e) ca:

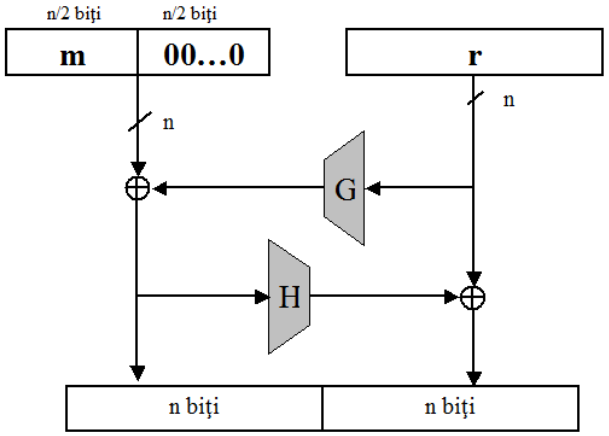
$$(OAEP(m, r))^e \mod N$$

- RSA-OEAP decriptează c folosind cheia secretă (N, d) ca:

$$(m, r) = OAEP^{-1}(c^d \mod N)$$

OAEP

- OAEP este definit astfel:



Notații

- G, H = funcții hash
- $m \in \{0,1\}^{n/2}$ mesajul
- $r \leftarrow^R \{0,1\}^n$
- se obține $OAEP(m, r)$ pe $2n$ biți

Important de reținut!

- Utilizarea RSA în practică:
  PKCS #1 v1.5 și PKCS #1 v2.0 (OAEP)

Securitatea Sistemelor Informatice

- Curs 10.2 -
Problema logaritmului discret

Adela Georgescu

Facultatea de Matematică și Informatică
Universitatea din București
Anul universitar 2022-2023, semestrul I

Algoritmi pentru calculul logaritmului discret

- Reamintim PLD:
- Fie $\mathbb{G}$ un grup ciclic de ordin q (cu $|q| = n$) iar g este generatorul lui $\mathbb{G}$.
- Pentru fiecare $h \in \mathbb{G}$ există un unic $x \in \mathbb{Z}_q$ a.î. $g^x = h$.
- PLD cere găsirea lui x știind $\mathbb{G}, q, g, h$; notăm $x = \log_g h$;
- Atenție! Atunci când $g^{x'} = h$ pentru un x' arbitrar (deci NU neapărat $x \in \mathbb{Z}_q$), notăm $log_g h = [x' \bmod q]$

Algoritmi pentru calculul logaritmului discret

- Problema PLD se poate rezolva, desigur, prin forță brută, calculând pe rând toate puterile x ale lui g până când se găsește una potrivită pentru care $g^x = h$;
- Complexitatea timp este $\mathcal{O}(q)$ iar complexitatea spațiu este $\mathcal{O}(1)$;
- Dacă se precalculează toate valorile (x, g^x), căutarea se face în timp $\mathcal{O}(1)$ și spațiu $\mathcal{O}(q)$;
- Sunt de interes algoritmii care pot obține un timp mai bun la rulare decât forța brută, realizând un compromis spațiu-timp.

Algoritmi pentru calculul logaritmului discret

- Se cunosc mai mulți astfel de algoritmi împărțiți în două categorii:
  - algoritmi *generici* care funcționează în grupuri arbitrare (i.e. orice grupuri ciclice);
  - algoritmi *non-generici* care lucrează în grupuri *specifice* - exploatează proprietăți speciale ale anumitor grupuri
- Dintre algoritmii generici enumerăm:
- Metoda **Baby-step/giant-step**, datorată lui Shanks, calculează logaritmul discret într-un grup de ordin q în timp $\mathcal{O}(\sqrt{q} \cdot (\log q)^c)$;
- pentru $g \in mathbbG$ generator, elementele lui $\mathbb{G}$ sunt

$$1 = g^0, g^1, g^2, ..., g^{q-1}, g^q = 1$$

- știm că $h = g^x$ se află între aceste valori

Metoda Baby-step/giant-step

- marcam și memorăm anumite puncte din grup, aflate la distanța $t = \lfloor \sqrt{q} \rfloor$ (*giant-steps*)

$$g^0, g^t, g^{2t}, ..., g^{\lfloor q/t \rfloor \cdot t}$$

- știm că $h = g^x$ se află în unul din aceste intervale
- calculand, cu *baby-steps*, valorile

$$h \cdot g^1, h \cdot g^2, ..., h \cdot g^t$$

- una din ele va fi egala cu unul din punctele marcate i.e. $h \cdot g^i = g^{k \cdot t}$
- Complexitatea timp este $\mathcal{O}(\sqrt{q} \cdot polylog(q))$ iar complexitatea spațiu este $\mathcal{O}(\sqrt{q})$

Algoritmi generici pentru calculul logaritmului discret

- Metoda Baby-Step/Giant-Step este optimă ca timp de rulare, însă există alți algoritmi mai eficienți d.p.d.v. al complexității spațiu;
- Algoritmul Pohlig-Hellman poate fi folosit atunci când se cunoaște factorizarea ordinului q al grupului iar timpul de rulare depinde de factorii primi ai lui q;
- Pentru ca algoritmul să nu fie eficient, trebuie ca cel mai mare factor prim al lui q să fie de ordinul 2^{160}.

Algoritmi non-generici pentru calculul logaritmului discret

- Algoritmii non-generici sunt potențial mai puternici decât cei generici;
- Cel mai cunoscut algoritm pentru PLD în $\mathbb{Z}_p^*$ cu p prim este algoritmul GNFS (General Number Field Sieve) cu complexitate timp $2^{\mathcal{O}(n^{1/3}\cdot(\log n)^{2/3})}$ unde $|p| = \mathcal{O}(n)$;
- Există și un alt algoritm non-generic numit *metoda de calcul a indicelui* care rezolvă DLP în grupuri ciclice $\mathbb{Z}_p^*$ cu p prim în timp sub-expoențial în lungimea lui p.
- Această metodă seamănă cu algoritmul sitei pătratice pentru factorizare;

Important de reținut!

- Cel mai bun algoritm pentru DLP este sub-exponențial;
- Se pot construi funcții hash rezistente la coliziuni bazate pe dificultatea DLP;

Securitatea Sistemelor Informatice

- Curs 10.3 -
Schimbul de chei Diffie-Hellman

Adela Georgescu

Facultatea de Matematică și Informatică
Universitatea din București
Anul universitar 2022-2023, semestrul I

Primitive cu cheie publică

- Am văzut că bazele criptografiei cu cheie publică au fost puse de Diffie și Hellman în 1976 ...
- ... când au introdus 3 primitive cu cheie publică diferite:
  1. **sisteme de criptare cu cheie publică**
  2. **semnături digitale**
  3. **schimb de chei**
- Deși au introdus 3 concepte diferite, Diffie și Hellman au introdus o singură construcție, pentru schimbul de chei.

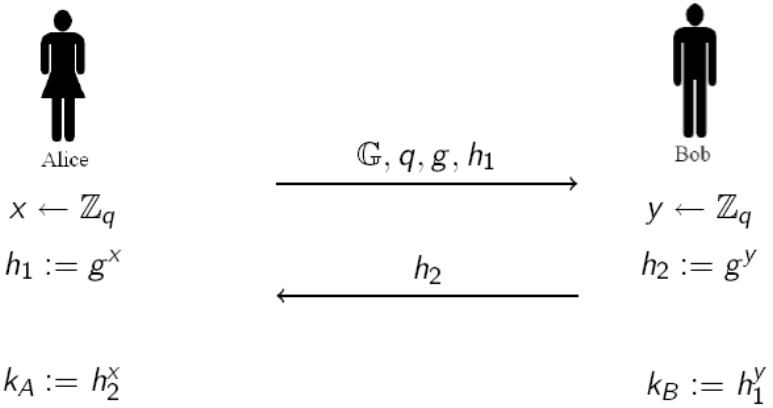
Schimb de chei

- **Sistemele de criptare cu cheie publică** le-am studiat și le vom mai studia în detaliu;
- **Semnăturile digitale** sunt analogul MAC-urilor din criptografia simetrică (sau corespondentul digital unei semnături reale);
- **Schimbul de chei** îl introducem pentru a facilita introducerea sistemelor de criptare bazate pe DLP.

Schimb de chei

- Un protocol de schimb de chei este un protocol prin care 2 persoane care nu partajează în prealabil nici un secret pot genera o cheie comună, secretă;
- Comunicarea necesară pentru stabilirea cheii se realizează printr-un canal public!
- Deci un adversar poate intercepta toate mesajele transmise pe canalul de comunicație, dar NU trebuie să afle nimic despre cheia secretă obținută în urma protocolului;
- Protocoalele de schimb de chei reprezintă o primitivă fundamentală în criptografie;
- În continuare, ne vom rezuma strict la schimbul de chei Diffie-Hellman.

Schimbul de chei Diffie-Hellman



Schimbul de chei Diffie-Hellman

- Alice și Bob doresc să stabilească o cheie secretă comună;
- Alice generează un grup ciclic $\mathbb{G}$, de ordin q cu $|q| = n$ și g un generator al grupului;
- Alice alege $x \leftarrow^R \mathbb{Z}_q$ și calculează $h_1 := g^x$;
- Alice îi trimite lui Bob mesajul $(\mathbb{G}, g, q, h_1)$;
- Bob alege $y \leftarrow^R \mathbb{Z}_q$ și calculează $h_2 := g^y$;
- Bob îi trimite h_2 lui Alice și întoarce cheia $k_B := h_1^y$;
- Alice primește h_2 și întoarce cheia $k_A = h_2^x$.

Schimbul de chei Diffie-Hellman

- Corectitudinea protocolului presupune ca $k_A = k_B$, ceea ce se verifică ușor:
- Bob calculează cheia

$$k_B = h_1^y = (g^x)^y = g^{xy}$$

- Alice calculează cheia

$$k_A = h_2^x = (g^y)^x = g^{xy}$$

Securitate

- O condiție **minimală** pentru ca protocolul să fie sigur este ca DLP să fie dificilă în $\mathbb{G}$;
- Întrebare: Cum poate un adversar pasiv să determine cheia comună dacă poate sparge DLP?
- Răspuns: Ascultă mediul de comunicație și preia mesajele h_1 și h_2. Rezolvă *DLP* pentru h_1 și determină x, apoi calculează $k_A = k_B = h_2^x$.
- Aceasta nu este însă singura condiție necesară pentru a proteja protocolul de un atacator pasiv!

CDH (Computational Diffie-Hellman)

- O condiție mai potrivită ar fi că adversarul să nu poată determina cheia comună $k_A = k_B$, chiar dacă are acces la întreaga comunicație;
- Aceasta este problema de calculabilitate Diffie-Hellman (CDH): Fiind date grupul ciclic $\mathbb{G}$, un generator g al său și 2 elemente $h_1, h_2 \leftarrow^R \mathbb{G}$, să se determine:

$$CDH(h_1, h_2) = g^{log_g h_1 log_g h_2}$$

- Pentru schimbul de chei Diffie-Hellman, rezolvarea CDH înseamnă că adversarul determină $k_A = k_B = g^{xy}$ cunoscând $h_1, h_2, \mathbb{G}, g$ (toate disponibile pe mediul de transmisiune nesecurizat).

DDH (Decisional Diffie-Hellman)

- Nici această condiție nu este suficientă: chiar dacă adversarul nu poate determina cheia exactă, poate de exemplu să determine părți din ea;
- O condiție și mai potrivită este ca pentru adversar, cheia $k_A = k_B$ să fie **indistinctibilă** față de o valoare aleatoare;
- Sau, altfel spus, să satisfacă problema de decidabilitate Diffie-Hellman (DDH):

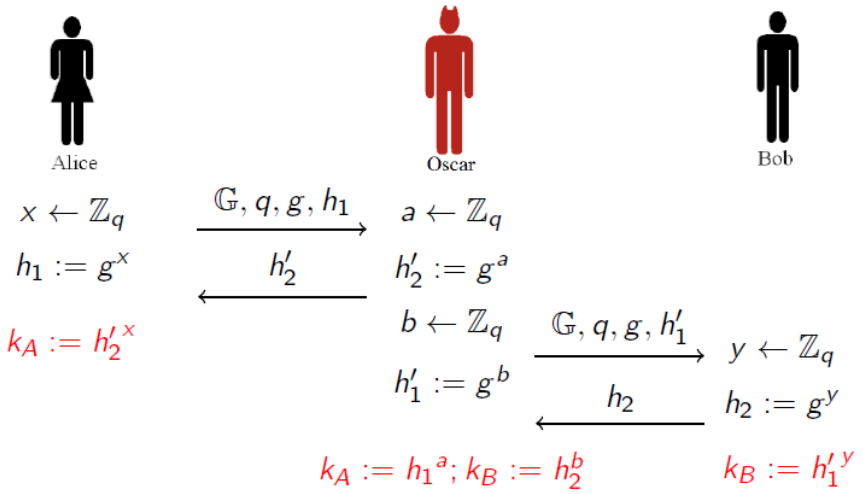
Definiție

Spunem că problema decizională Diffie-Hellman (DDH) este dificilă (relativ la $\mathbb{G}$), dacă pentru orice algoritm PPT $\mathcal{A}$ există o funcție neglijabilă negl *așa încât:*
$|Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^z) = 1] - Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = 1]| \leq negl(n)$,
unde $x, y, z \leftarrow^R \mathbb{Z}_q$

Atacul Man-in-the-Middle

- Am analizat până acum securitatea față de atacatori pasivi;
- Arătăm acum că schimbul de chei Diffie-Hellman este total nesigur pentru un adversar activ ...
- ... care are dreptul de a intercepta, modifica, elimina mesajele de pe calea de comunicație;
- Un astfel de adversar se poate interpune între Alice și Bob, dând naștere unui atac de tip Man-in-the-Middle.

Atacul Man-in-the-Middle



Atacul Man-in-the-Middle

- Alice generează un grup ciclic $\mathbb{G}$, de ordin q cu $|q| = n$ și g un generator al grupului;
- Alice alege $x \leftarrow^R \mathbb{Z}_q$ și calculează $h_1 := g^x$;
- Alice îi trimite lui Bob mesajul $(\mathbb{G}, g, q, h_1)$;
- Oscar interceptează mesajul și răspunde lui Alice în locul lui Bob: alege $a \leftarrow^R \mathbb{Z}_q$ și calculează $h'_2 := g^a$;
- Oscar și Alice dețin acum cheia comună $k_A = g^{xa}$;
- Oscar inițiază, în locul lui Alice, o nouă sesiune cu Bob: alege $b \leftarrow^R \mathbb{Z}_q$ și calculează $h'_1 := g^b$;
- Bob alege $y \leftarrow^R \mathbb{Z}_q$ și calculează $h_2 := g^y$;
- Oscar și Bob dețin acum cheia comună $k_B = g^{yb}$.

Atacul Man-in-the-Middle

- Atacul este posibil pentru că Oscar poate **impersona** pe Alice și pe Bob;
- De fiecare dată când Alice va transmite un mesaj criptat către Bob, Oscar îl interceptează și îl previne să ajungă la Bob;
- Oscar îl decriptează folosind k_A, apoi îl recriptează folosind k_B și îl transmite către Bob;
- Alice și Bob comunică fără să fie conștienți de existența lui Oscar.

Important de reținut!

- Schimbul de chei - o primitivă cripografică importantă
- Prezumții criptografice: CDH, DDH
- Schimbul de chei Diffie-Hellman nu rezistă la atacuri active

Securitatea Sistemelor Informatice

- Curs 10.4 -
ElGamal

Adela Georgescu

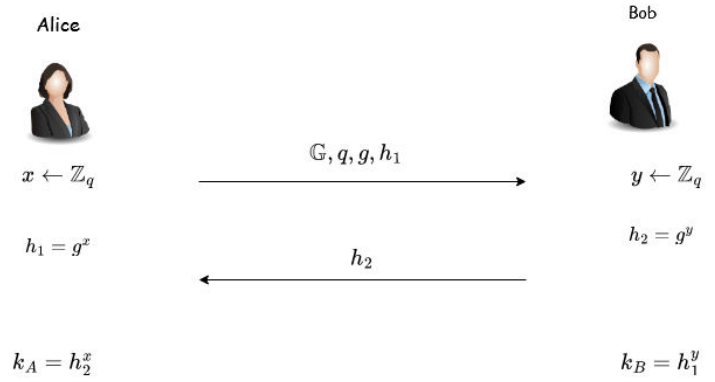
Facultatea de Matematică și Informatică
Universitatea din București
Anul universitar 2022-2023, semestrul I

Sistemul de criptare ElGamal

- 1976 - Diffie și Hellman definesc conceptul de criptografie asimetrică;
- 1977 - R.Rivest, A.Shamir și Leonard Adleman introduc sistemul RSA;
- 1985 - T.ElGamal propune un nou sistem de criptare.

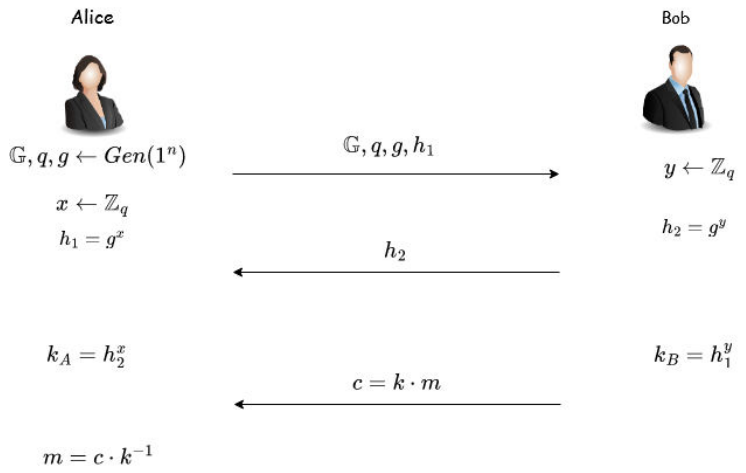
Sistemul de criptare ElGamal

- Reamintim schimbul de chei Diffie-Hellman



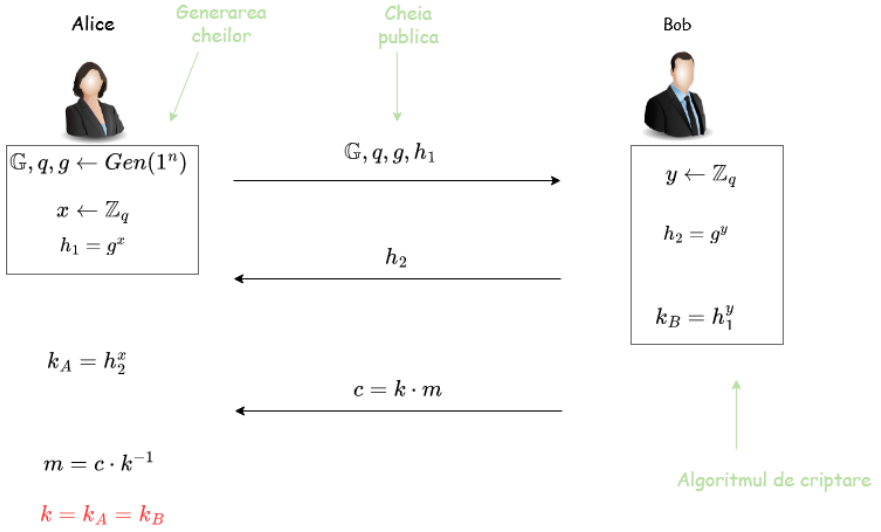
Sistemul de criptare ElGamal

- Il modificăm așa încât să permită criptare

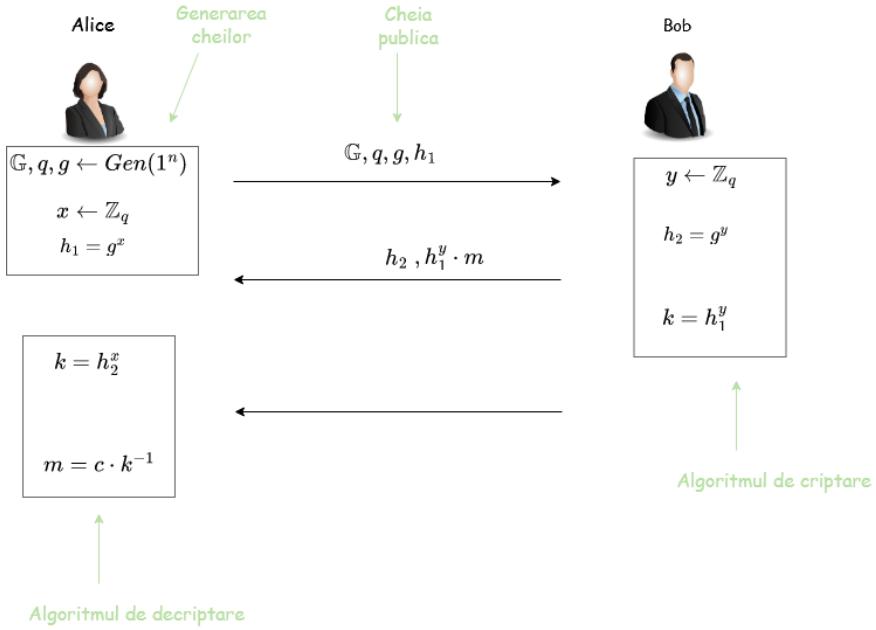


Sistemul de criptare ElGamal

- Acum poate fi văzut ca un sistem de criptare



Sistemul de criptare ElGamal



Sistemul de criptare ElGamal

- Definim sistemul de criptare *ElGamal* pe baza ideii prezentate anterior;
  1. Se generează $(\mathbb{G}, q, g)$, se alege $x \leftarrow^R \mathbb{Z}_q$ și se calculează $h = g^x$;
     - Cheia publică este: $(\mathbb{G}, q, g, h)$;
     - Cheia privată este x;
  2. **Enc**: dată o cheie publică $(\mathbb{G}, q, g, h)$ și un mesaj $m \in \mathbb{G}$, alege $y \leftarrow^R \mathbb{Z}_q$ și întoarce $c = (c_1, c_2) = (g^y, m \cdot h^y)$;
  3. **Dec**: dată o cheie secretă $(\mathbb{G}, q, g, x)$ și un mesaj criptat $c = (c_1, c_2)$, întoarce $m = c_2 \cdot c_1^{-x}$.

Securitate - Problema 1

Problema 1: Determinismul

- Întrebare: Este sistemul ElGamal determinist?
- Răspuns: NU! Sistemul este nedeterminist, datorită alegerii aleatoare a lui y la fiecare criptare.
- Un același mesaj m se poate cripta diferit, pentru $y \neq y'$:

$$c = (c_1, c_2) = (g^y, m \cdot h^y)$$

$$c' = (c_1', c_2') = (g^{y\prime}, m \cdot h^{y\prime})$$

Securitate - Problema 2

Problema 2: Dificultatea DLP

- Întrebare: Rămâne ElGamal sigur dacă problema DLP este simplă?
- Răspuns: NU! Se determină x a.î. $h = g^x$, apoi se decriptează orice mesaj pentru că se cunoaște cheia secretă.

Securitate - Problema 3

Problema 3: Proprietatea de homomorfism

- Fie m_1, m_2 2 texte clare și $c_1 = (c_{11}, c_{12}), c_2 = (c_{21}, c_{22})$ textele criptate corespunzătoare;
- Atunci:
  $c_1 \cdot c_2 = (c_{11} \cdot c_{21}, c_{12} \cdot c_{22}) = (g^{y_1} \cdot g^{y_2}, m_1 h^{y_1} \cdot m_2 h^{y_2})$
- Întrebare: Dacă un adversar cunoaște c_1 și c_2 criptările lui m_1, respectiv m_2, ce poate spune despre $c_1 \cdot c_2$?
- Răspuns: $c_1 \cdot c_2$ este criptarea lui $m_1 \cdot m_2$ folosind $y = y_1 + y_2$:
  $c_1 \cdot c_2 = (g^{y_1+y_2}, m_1 m_2 h^{y_1+y_2})$
- Un sistem de criptare care satisface $Dec_s k(c_1 \cdot c_2) = Dec_{sk}(c_1) \cdot Dec_{sk}(c_2)$ se numește sistem de criptare **homomorfic**.
  (homomorfismul este deseori o proprietate utilă în criptografie)

Securitate - Problema 4

Problema 4: Utilizarea multiplă a parametrilor publici

- Este comun în practică pentru un administrator să fixeze parametrii publici $(\mathbb{G}, q, g)$, apoi fiecare utilizator să își genereze doar cheia secretă x și să publice $h = g^x$;
- Întrebare: Este corect să se utilizeze de mai multe ori aceiași parametrii publici $(\mathbb{G}, q, g)$?
- Răspuns: Se consideră că DA. Cunoașterea parametrilor publici pare să nu conducă la rezolvarea DDH.
- Atenție! Acest lucru nu se întâmpla și la RSA, unde modulul NU trebuie utilizat de mai multe ori.

Securitate - teoremă

Teoremă

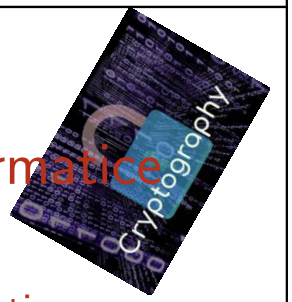
Dacă problema decizională Diffie-Hellman (DDH) este dificilă în grupul $\mathbb{G}$, atunci schema de criptare ElGamal este CPA-sigură.

- Se poate vedea din securitatea schimbului de chei Diffie-Hellman
- In forma aceasta, sistemul ElGamal nu este CCA-sigur...pentru ca este maleabil
  Insă poate fi modificat așa încât să fie CCA-sigur

Important de reținut!

- Sistemul de criptare ElGamal
- Proprietatea de homomorfism

Securitatea Sistemelor Informatice



- Curs 10.5 -
Criptografia bazată pe curbe eliptice

Adela Georgescu

Facultatea de Matematică și Informatică
Universitatea din București
Anul universitar 2022-2023, semestrul I

Grupuri ciclice pentru uz criptografic

- În cursul precendent am discutat despre grupuri ciclice și am subliniat faptul că sunt de preferat, pentru criptografie, grupurile ciclice de ordin prim;
- Am menționat că, de obicei, se lucrează în grupul $\mathbb{Z}_p^*$ cu p prim iar un subgrup de ordin prim al lui este format din mulțimea resturilor pătratice modulo p;
- În continuare introducem o altă clasă de grupuri care constă din punctele unei curbe eliptice;
- Aceste grupuri sunt folosite în criptografie pentru că, spre deosebire de $\mathbb{Z}_p^*$, nu se cunoaște deocamdată nici un algoritm sub-exponențial pentru rezolvarea DLP în aceste grupuri.

Curbe eliptice

Definiție

O curbă eliptică peste $\mathbb{Z}_p$, $p > 3$ prim, este mulțimea perechilor (x, y) cu $x, y \in \mathbb{Z}_p$ așa încât

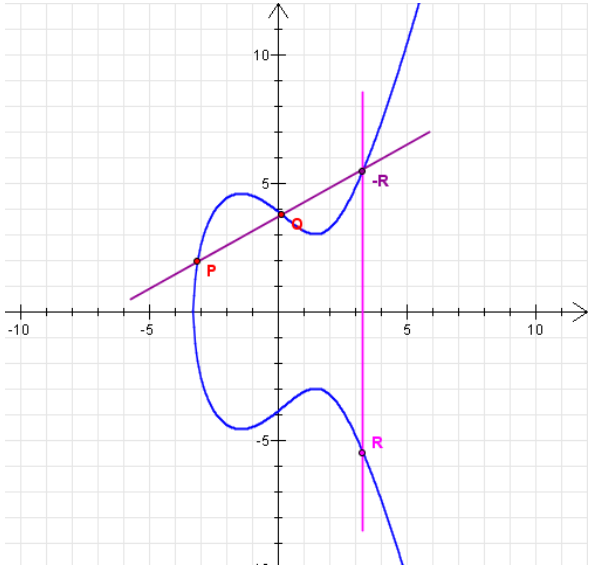
$$y^2 = x^3 + Ax + B \bmod p$$

împreună cu punctul la infinit $\mathcal{O}$ unde
$A, B \in \mathbb{Z}_p$ sunt constante care respectă $4A^3 + 27B^2 \neq 0 \bmod p$

- Vom nota cu $E(\mathbb{Z}_p)$ o curbă eliptică definită peste $\mathbb{Z}_p$

Curbe eliptice

O curbă eliptică peste spațiul numerelor reale $\mathbb{R}$
$E(\mathbb{R}) : y^2 = x^3 - x + 1$



Grupul punctelor de pe o curba eliptică

- Pentru a arăta că punctele de pe o curbă eliptică formează un grup ciclic, definim o operație de grup peste aceste puncte:
- Definim operația binară aditivă "+" astfel:
  - punctul la infinit $\mathcal{O}$ este element neutru: $\forall P \in E(\mathbb{Z}_p)$ definim

$$P + \mathcal{O} = \mathcal{O} + P = P.$$

  - fie $P = (x_1, y_1)$ și $Q = (x_2, y_2)$ două puncte de pe curbă; atunci:
  - dacă $x_1 = x_2$ și $y_2 = -y_1$, $P + Q = \mathcal{O}$

Grupul punctelor de pe o curba eliptică

- altfel, P + Q = R de coordonate (x_3, y_3) care se calculează astfel:

$$x_3 = [m^2 - x_1 - x_2 \bmod p]$$
$$y_3 = [m(x_1 - x_3) - y_1 \bmod p]$$

- iar m se calculează astfel:

$$m = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} \bmod p & \text{dacă } P \neq Q \\ \frac{3x_1^2 + A}{2y_1} \bmod p & \text{dacă } P = Q \end{cases}$$

- dacă $P = Q$ si $y_1 = 0$, atunci $P + Q = 2P = \mathcal{O}$

Grupul punctelor de pe o curba eliptică

- Geometric, suma a două puncte P si Q se obține trasând o linie prin cele două puncte și găsind cel de-al trelea punct R de intersecție al liniei cu E;
- În această situație, m reprezintă panta dreptei care trece prin P și Q;
- Se poate arăta că mulțimea de puncte $E(\mathbb{Z}_p)$ împreună cu operația aditivă definită formează un grup abelian;
- Există o teoremă de structură pentru $E(\mathbb{Z}_p)$ care exprimă condițiile în care grupul este ciclic.

Grupul punctelor de pe o curba eliptică

- In practică, sunt căutate acele curbe eliptice pentru care ordinul grupului ciclic generat este prim;
- Pentru criptografie, sunt de interes curbe eliptice de ordin mare
- O curbă eliptică definită peste $\mathbb{Z}_p$ are aproximativ p puncte. Mai precis [Teorema lui Hasse]:

$$p + 1 - 2\sqrt{p} \leq card(E(\mathbb{Z}_p)) \leq p + 1 + 2\sqrt{p}$$

- Există mai multe clase de curbe eliptice slabe d.p.d.v. criptografic, iar ele trebuiesc evitate.
- De pildă, curbe eliptice peste $\mathbb{Z}_p$ cu $card(E(\mathbb{Z}_|)) = p$
- In practică, se folosesc curbe eliptice standardizate

Curbe eliptice folosite în practică

Curbe eliptice standardizate, folosite în practică, sigure și cu implementări eficiente:

- *curba eliptică* P-256 (sau $secp256r1$) este o curbă eliptică peste $\mathbb{Z}_p$ cu p pe 256 biți de forma $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$. Această curbă are ecuația $y^2 = x^3 - 3x + B \mod p$ iar p-ul ales astfel permite o implementare eficientă. Curbele P-384 ($secp384r1$) și P-521 ($secp521r1$) sunt definite in mod analog
- *curba eliptică* 25519 este definită peste $\mathbb{Z}_p$ cu p pe 255 biți de forma $p = 2^{255} - 19$ și permite o implementare eficientă. Grupul acestei curbe eliptice nu are ordin prim dar se poate lucra într-un subgrup de ordin mare prim

Curbe eliptice folosite în practică

Curbe eliptice standardizate, folosite în practică, sigure și cu implementări eficiente:

- $secp256k1$ este o curbă eliptică de ordin prim peste $\mathbb{Z}_p$ cu p pe 256 biți de forma $p = 2^{256} - 2^{232} - 2^{29} - 2^{28} - 2^{27} - 2^{26} - 2^{24} - 1$ și are ecuatia $y^2 = x^3 + 7 \mod p$. Aceasta curbă eliptică este folosită în Bitcoin.

ECDLP - Problema logaritmului discret pe curbe eliptice

- ECDLP = Elliptic Curve Discrete Logarithm Problem
- Putem defini acum DLP în grupul punctelor unei curbe eliptice (ECDLP):
- Fie E o curbă eliptică peste $\mathbb{Z}_p$, un punct $P \in \mathbb{Z}_p$ de ordin n și Q un element din subgrupul ciclic generat de P:

$$Q \in [P] = \{sP \mid 1 \leq s \leq n-1\}$$

- Problema ECDLP cere găsirea unui k așa încât $Q = kP$;
- Notație: $\underbrace{P + P + ... + P}_{s \text{ ori}} = sP$.

ECDLP - Securitate

- Alegând cu grijă curbele eliptice, cel mai bun algoritm pentru rezolvarea ECDLP este considerabil mai slab decât cel mai bun algoritm pentru rezolvarea problemei DLP în $\mathbb{Z}_p^*$;
- De exemplu, algoritmul de calcul al indicelui nu este deloc eficient pentru ECDLP;
- Pentru anumite curbe eliptice, singurii algoritmi de rezolvare sunt algoritmii generici pentru DLP, adică metoda baby-step giant-step și metoda Pollard rho;
- Cum numărul de pași necesari pentru un astfel de algoritm este de ordinul rădăcinii pătrate a cardinalului grupului, se recomandă folosirea unui grup de ordin cel puțin 2^{160}.

ECDLP - Securitate

- O consecință a teoremei lui Hasse este că dacă avem nevoie de o curbă eliptică cu 2^{160} elemente, trebuie să folosim un număr prim p pe aproximativ 160 biți;
- Deci, dacă folosim o curbă eliptică $E(\mathbb{Z}_p)$ cu p pe 160 biți, un atac generic asupra ECDLP are 2^{80} complexitate timp;
- Un nivel de securitate de 80 biți oferă securitate pe termen mediu;
- În practică, curbe eliptice peste $\mathbb{Z}_p$ cu p până la 256 biți sunt folosite, cu un nivel de securitate pe 128 biți.

Criptografia pe curbe eliptice - ECC (Elliptic Curve Cryptography)

- A fost inventată independent în 1987 de Neal Koblitz și în 1986 de Victor Miller;
- La începutul anilor 1990 se făceau foarte multe speculații despre securitatea și practicalitatea ECC, mai ales comparativ cu RSA;
- După cercetări intensive, ECC pare foarte sigură, la fel de sigură precum RSA sau schemele bazate pe DLP;
- Încrederea a crescut după ce în 1999 și 2001 au fost standardizate, pentru domeniul bancar, semnături digitale și schimburi de chei bazate pe curbe eliptice.

Criptografia pe curbe eliptice - ECC (Elliptic Curve Cryptography)

- Curbele eliptice sunt folosite pe larg și în standardele comerciale precum IPsec sau TLS;
- ECC este adesea preferată în fața criptografiei cu cheie publică pentru sistemele încorporate precum dispozitivele mobile...
- ...din motive de performanță;
- implementările pentru ECC sunt considerabil mai mici și mai rapide decât cele pentru RSA;
- ECC cu chei pe 160-250 biți oferă cam același nivel de securitate precum RSA sau sistemele bazate pe DLP cu chei pe 1024-3072 biți.

Comparație între ECC, criptografia simetrică și asimetrică

Algorithm Family	Cryptosystems	Security Level (bit)			
		80	128	192	256
Integer factorization	RSA	1024 bit	3072 bit	7680 bit	15360 bit
Discrete logarithm	DH, DSA, Elgamal	1024 bit	3072 bit	7680 bit	15360 bit
Elliptic curves	ECDH, ECDSA	160 bit	256 bit	384 bit	512 bit
Symmetric-key	AES, 3DES	80 bit	128 bit	192 bit	256 bit

Figure: [Understanding cryptography, Christoph Paar, Jan Pelzl, Springer 2010]

- Un algoritm are nivelul de securitate "security level" pe n biți dacă cel mai bun atac necesită 2^n pași.

Important de reținut!

- Curbele eliptice oferă un suport bun pentru criptografie;
- ECDLP este dificilă.

Securitatea Sistemelor Informatice

- Curs 10.6 -
Schimbul de chei Diffie-Hellman și ElGamal pe curbe eliptice

Adela Georgescu

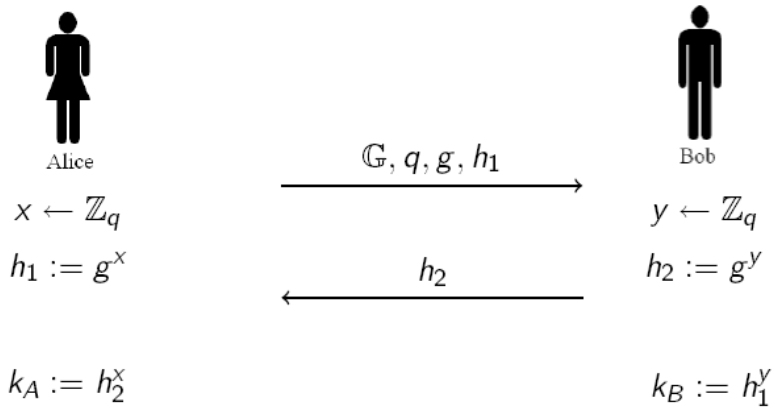
Facultatea de Matematică și Informatică
Universitatea din București
Anul universitar 2022-2023, semestrul I

Schimbul de chei Diffie-Hellman pe curbe eliptice

- Am studiat schimbul de chei Diffie-Hellman peste un grup ciclic $\mathbb{G}$, de ordin q;
- Transpunem construcția pe curbe eliptice:

$$(\mathbb{G}, \cdot) \to (E(\mathbb{Z}_q), +)$$

Schimbul de chei Diffie-Hellman pe curbe eliptice



Schimbul de chei Diffie-Hellman pe curbe eliptice

- Alice și Bob doresc să stabilească o cheie secretă comună;
- Alice generează o curbă eliptică $E(\mathbb{Z}_q)$, și P un punct pe curbă (generator);
- Alice alege $x \leftarrow^R \mathbb{Z}_q$ și calculează $H_1 := xP$;
- Alice îi trimite lui Bob mesajul $(E(\mathbb{Z}_q), P, H_1)$;
- Bob alege $y \leftarrow^R \mathbb{Z}_q$ și calculează $H_2 := yP$;
- Bob îi trimite H_2 lui Alice și întoarce cheia $k_B := yH_1$;
- Alice primește H_2 și întoarce cheia $k_A = xH_2$.

Schimbul de chei Diffie-Hellman pe curbe eliptice

- Corectitudinea protocolului presupune ca $k_A = k_B$, ceea ce se verifică ușor:
- Bob calculează cheia

$$k_B = yH_1 = y(xP) = (xy)P$$

- Alice calculează cheia

$$k_A = xH_2 = x(yP) = (xy)P$$

Securitate

- O condiție **minimală** pentru ca protocolul să fie sigur este ca ECDLP să fie dificilă în $\mathbb{G}$;
- Întrebare: Cum poate un adversar pasiv să determine cheia comună dacă poate sparge ECDLP?
- Răspuns: Ascultă mediul de comunicație și preia mesajele H_1 și H_2. Rezolvă *ECDLP* pentru H_1 și determină x, apoi calculează $k_A = k_B = xH_2$.
- Aceasta nu este însă singura condiție necesară pentru a proteja protocolul de un atacator pasiv!

ECCDH (Elliptic Curve Computational Diffie-Hellman)

- O condiție mai potrivită ar fi că adversarul să nu poată determina cheia comună $k_A = k_B$, chiar dacă are acces la întreaga comunicație;
- Aceasta este problema de calculabilitate Diffie-Hellman pe curbe eliptice (ECCDH): Fiind date curba eliptică $E(\mathbb{Z}_q)$, un punct P pe curbă și 2 alte puncte $H_1, H_2 \leftarrow^R E(\mathbb{Z}_q)$, să se determine:

$$ECCDH(H_1, H_2) = (ECDLP(P, H_1)ECDLP(P, H_2))P$$

- Pentru schimbul de chei Diffie-Hellman, rezolvarea ECCDH înseamnă că adversarul determină $k_A = k_B = xyP$ cunoscând $H_1, H_2, E(\mathbb{Z}_q), P$ (toate disponibile pe mediul de transmisiune nesecurizat).

ECDDH (Elliptic Curve Decisional Diffie-Hellman)

- Nici această condiție nu este suficientă: chiar dacă adversarul nu poate determina cheia exactă, poate de exemplu să determine părți din ea;
- O condiție și mai potrivită este ca pentru adversar, cheia $k_A = k_B$ să fie **indistinctibilă** față de o valoare aleatoare;
- Sau, altfel spus, să satisfacă problema de decidabilitate Diffie-Hellman pe curbe eliptice (ECDDH):

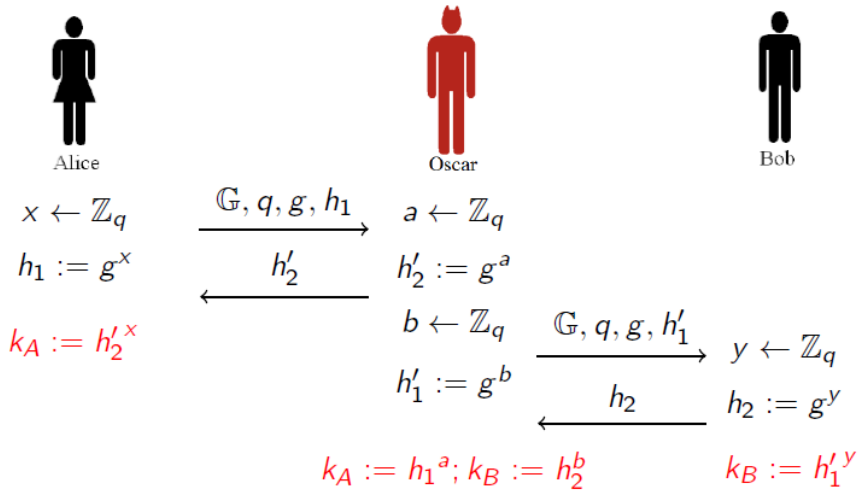
Definiție

Spunem că problema decizională Diffie-Hellman (ECDDH) este dificilă (relativ la curba eliptică $E(\mathbb{Z}_q)$), dacă pentru orice algoritm PPT $\mathcal{A}$ există o funcție neglijabilă negl *așa încât:*
$|Pr[\mathcal{A}(E(\mathbb{Z}_q), P, xP, yP, zP) = 1]$ - $Pr[\mathcal{A}(E(\mathbb{Z}_q), P, xP, yP, xyP) = 1]|$
$\leq negl(n)$, *unde* $x, y, z \leftarrow^R \mathbb{Z}_q$

Atacul Man-in-the-Middle

- Am analizat până acum securitatea față de atacatori pasivi;
- Arătăm acum că schimbul de chei Diffie-Hellman este total nesigur pentru un adversar activ ...
- ... care are dreptul de a intercepta, modifica, elimina mesajele de pe calea de comunicație;
- Un astfel de adversar se poate interpune între Alice și Bob, dând naștere unui atac de tip Man-in-the-Middle.

Atacul Man-in-the-Middle



Atacul Man-in-the-Middle

- Alice generează o curbă eliptică $E(\mathbb{Z}_q)$ și P un punct pe curbă;
- Alice alege $x \leftarrow^R \mathbb{Z}_q$ și calculează $H_1 := xP$;
- Alice îi trimite lui Bob mesajul $(E(\mathbb{Z}_q), P, H_1)$;
- Oscar interceptează mesajul și răspunde lui Alice în locul lui Bob: alege $a \leftarrow^R \mathbb{Z}_q$ și calculează $H_2' := aP$;
- Oscar și Alice dețin acum cheia comună $k_A = axP$;
- Oscar inițiază, în locul lui Alice, o nouă sesiune cu Bob: alege $b \leftarrow^R \mathbb{Z}_q$ și calculează $H_1' := bP$;
- Bob alege $y \leftarrow^R \mathbb{Z}_q$ și calculează $H_2 := yP$;
- Oscar și Bob dețin acum cheia comună $k_B = ybP$.

Atacul Man-in-the-Middle

- Atacul este posibil pentru că poate **impersona** pe Alice și pe Bob;
- De fiecare dată când Alice va transmite un mesaj criptat către Bob, Oscar îl interceptează și îl previne să ajungă la Bob;
- Oscar îl decriptează folosind k_A, apoi îl recriptează folosind k_B și îl transmite către Bob;
- Alice și Bob comunică fără să fie conștienți de existența lui Oscar.

Sistemul de criptare ElGamal pe curbe eliptice

- Am studiat sistemul de criptare ElGamal peste un grup ciclic $\mathbb{G}$, de ordin q;
- Transpunem construcția pe curbe eliptice:

$$(\mathbb{G}, \cdot) \to (E(\mathbb{Z}_q), +)$$

Sistemul de criptare ElGamal pe curbe eliptice

1. Se generează $E(\mathbb{Z}_q)$ o curbă eliptică și P un punct pe curbă (generator), se alege $z \leftarrow^R \mathbb{Z}_q$ și se calculează $H = xP$;
   - Cheia publică este: $(E(\mathbb{Z}_q), P, H)$;
   - Cheia privată este $(E(\mathbb{Z}_q), P, x)$;
2. **Enc**: dată o cheie publică $(E(\mathbb{Z}_q), P, H)$ și un mesaj $M \in E(\mathbb{Z}_q)$, alege $y \leftarrow^R \mathbb{Z}_q$ și întoarce $C = (C_1, C_2) = (yP, M + yH)$;
3. **Dec**: dată o cheie secretă $(E(\mathbb{Z}_q), P, x)$ și un mesaj criptat $C = (C_1, C_2)$, întoarce $M = C_2 + x(-C_1)$.

Securitate

- Sistemul transpus pe curbe eliptice păstează proprietățile sistemului inițial;
- Deci curba eliptică trebuie aleasă a.î. ECDLP și ECDDH să fie dificile...
- ... și sistemul rămâne nedeterminist și homomorfic.

Important de reținut!

- Modalitatea de trecere de la o construcție peste $(\mathbb{Z}_q, \cdot)$ la $(E(\mathbb{Z}_q), +)$
- Prezumții criptografice: ECCDH, ECDDH
- Schimbul de chei Diffie-Hellman pe curbe eliptice păstrează proprietățile schimbului de chei Diffie Hellman definit peste $\mathbb{G}$ grup ciclic de ordin q

Securitatea Sistemelor Informatice

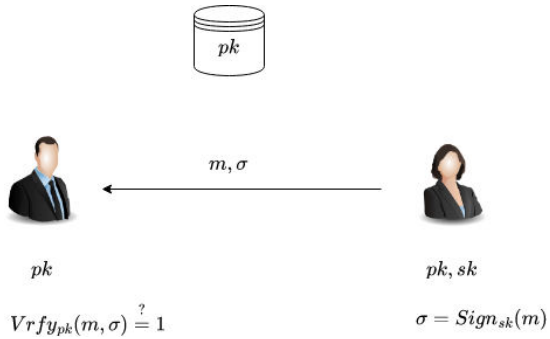
- Curs 11 -
Semnături digitale și PKI

Adela Georgescu

Facultatea de Matematică și Informatică
Universitatea din București
Anul universitar 2022-2023, semestrul I

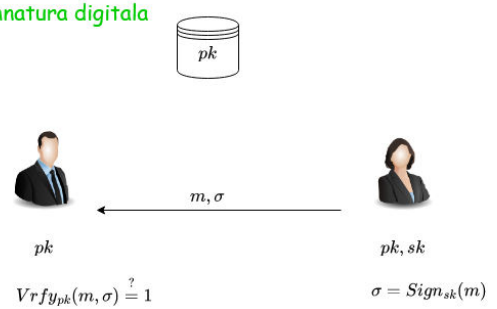
Scheme de semnătură digitală

- Schemele de semnătură digitală reprezintă echivalentul MAC-urilor în criptografia cu cheie publică, deși există câteva diferențe importante între ele;
- O schemă de semnătură digitală îi permite unui semnatar S care a stabilit o cheie publică *pk* să semneze un mesaj în așa fel încât oricine care cunoaște cheia *pk* poate verifica originea mesajului (ca fiind S) și integritatea lui;

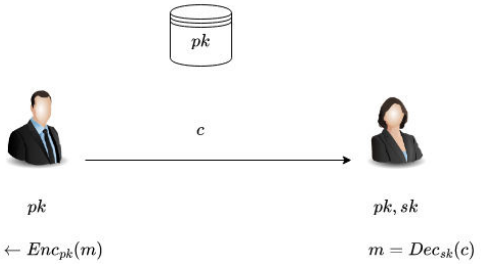


Semnătură digitală vs. criptare

Semnatura digitala

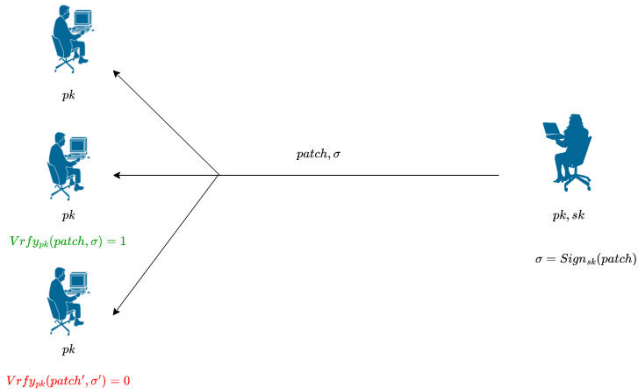


Criptare



Aplicații ale semnăturilor digitale

- De pildă, o companie de software vrea să transmită patch-uri de software într-o manieră autentificată, așa încât orice client să poată recunoaște dacă un patch e autentic;
- În schimb, o persoană malițioasă nu poate păcăli un client să accepte un patch care a nu a fost realizat de compania respectivă.

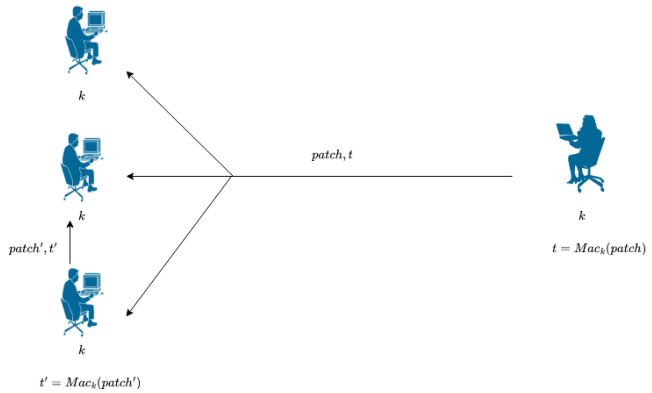


Aplicații ale semnăturilor digitale

- Pentru aceasta, compania generează o cheie publică *pk* împreună cu o cheie secretă *sk* și distribuie cheia *pk* clienților săi, păstrând cheia secretă;
- Atunci când lansează un patch de software *patch*, compania calculează o semnătură digitală σ pentru *patch* folosind cheia *sk* și trimite fiecărui client perechea $(patch, \sigma)$;
- Fiecare client stabilește autenticitatea lui *patch* verificând dacă σ este o semnătură legitimă pentru *patch* cu privire la cheia publică *pk*;
- Deci compania folosește aceeași cheie publică pentru toți clienții și calculează o singură semnătură pe care o trimite tuturor.

Aplicații ale semnăturilor digitale

- Putem înlocui semnătura digitală cu un MAC?



- In această situație, oricare dintre clienți poate emite un tag valid pentru un alt patch decât cel original și chiar îl poate trimite celorlalți clienți

Avantaje semnături digitale față de MAC-uri

- MAC-urile și schemele de semnătură digitală sunt folosite pentru asigurarea integrității (autenticității) mesajelor cu următoarele **diferențe**:
- Schemele de semnătură digitală sunt *public verificabile*...
- ...ceea ce înseamnă că semnăturile digitale sunt *transferabile* - o terță parte poate verifica legitimitatea unei semnături și poate face o copie pentru a convinge pe altcineva că aceea este o semnătură validă pentru m;
- Schemele de semnătură digitală au proprietatea de *non-repudiere* - un semnatar nu poate nega faptul că a semnat un mesaj;
- MAC-urile au avantajul că sunt cam de 2-3 ori mai eficiente (mai rapide) decât schemele de semnătură digitală.

Semnături digitale - Definiție

Definiție

O semnătură digitală definita peste $(\mathcal{K}, \mathcal{M}, \mathcal{S})$ *este formată din trei algoritmi polinomiali* $(\mathrm{Gen}, \mathrm{Sign}, \mathrm{Vrfy})$ *unde:*

1. Gen*: este algoritmul de generare a perechii de cheie publică și cheie privată* (pk, sk)
2. $\mathrm{Sign} : \mathcal{K} \times \mathcal{M} \to \mathcal{S}$ *este algoritmul de generare a semnăturilor* $\sigma \leftarrow \mathrm{Sign}_{sk}(m)$*;*
3. $\mathrm{Vrfy} : \mathcal{K} \times \mathcal{M} \times \mathcal{S} \to \{0, 1\}$
   este algoritmul de verificare ce întoarce un bit $b = \mathrm{Vrfy}_{pk}(m, \sigma)$ *cu semnificația că:*
   - $b = 1$ *înseamnă* valid
   - $b = 0$ *înseamnă* invalid

a.î : $\forall m \in \mathcal{M},\ k \in \mathcal{K},\ \mathrm{Vrfy}_{pk}(m, \mathrm{Sign}_{sk}(m)) = 1.$

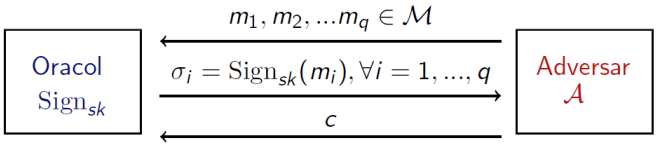
Securitate semnatură digitală - formalizare

- Formal, îi dăm adversarului acces la un *oracol* $\mathrm{Sign}_{sk}(\cdot)$;
- Adversarul poate trimite orice mesaj m dorit către oracol și primește înapoi o semnătură corespunzătoare $\sigma \leftarrow \mathrm{Sign}_{sk}(m)$;
- Considerăm că securitatea este impactată dacă adversarul este capabil să producă un mesaj m împreună cu o semnătură σ așa încât:
  1. σ este o semnătură validă pentru mesajul m: $\mathrm{Vrfy}_{pk}(m, \sigma) = 1$;
  2. Adversarul nu a solicitat anterior (de la oracol) o semnătură pentru mesajul m.

Securitate semnatură digitală - formalizare

- Despre o semnătură care satisface nivelul de securitate de mai sus spunem că *nu poate fi falsificată printr-un atac cu mesaj ales*;
- Aceasta înseamnă că un adversar nu este capabil să falsifice o semnătură validă pentru nici un mesaj ...
- ... deși poate obține semnături pentru orice mesaj ales de el, chiar *adaptiv* în timpul atacului.
- Pentru a da definiția formală, definim mai întâi un experiment pentru o semnătura $\pi = (\mathrm{Sign}, \mathrm{Vrfy})$, în care considerăm un adversar $\mathcal{A}$ și parametrul de securitate n;

Experimentul $\mathrm{Sign}^{forge}_{\mathcal{A},\pi}(n)$



- Adversarul întoarce o pereche de mesaj, semnătură (m, σ)
- Output-ul experimentului este 1 dacă și numai dacă: (1) $\mathrm{Vrfy}_{pk}(m, \sigma) = 1$ și (2) $m \notin \{m_1, ..., m_q\}$;
- Dacă $\mathrm{Sign}^{forge}_{\mathcal{A},\pi}(n) = 1$, spunem că $\mathcal{A}$ a efectuat experimentul cu succes.

Securitate semnaturi digitale

Definiție

O semnătură $\pi = (\mathrm{Gen}, \mathrm{Sign}, \mathrm{Vrfy})$ *este sigură (nu poate fi falsificată printr-un atac cu mesaj ales) dacă pentru orice adversar polinomial* $\mathcal{A}$ *există o funcție neglijabilă* $negl$ *așa încât*

$$Pr[\mathrm{Sign}^{forge}_{\mathcal{A},\pi}(n) = 1] \leq negl(n).$$

Atacuri prin replicare

- Un atacator poate prelua o pereche de mesaj și semnătură digitală și o retrimite unui destinatar
- la fel ca în criptografia simetrică, definiția semnăturilor digitale nu protejează împotriva acestui tip de atac
- in contextul exemplului cu patch-ul de software, acest atac este problematic

Construcție scheme de semnătură digitală

- Paradigma "hash-and-sign" este sigură: înainte de semnare, mesajul trece printr-o funcție hash; varianta aceasta se folosește pe larg în practică;
  - $\sigma \leftarrow \mathrm{Sign}_{sk}(H(m))$; $\mathrm{Vrfy}_{\mathrm{pk}}(\mathrm{H}(\mathrm{m}), \sigma) \stackrel{?}{=} 1$
- construcția este sigură atâta timp cât H este rezistentă la coliziuni
- este avantajoasă pentru că oferă funcționalitatea unei semnături digitale (criptografie cu cheie publică) la costul unei operații din criptografia cu cheie secretă
- folosită pe larg în practică

Prezumpția RSA - reamintire

- Se aleg două numere mari prime p și q
- Se calculează modulul $N = p \cdot q$
- Fie $\mathbb{Z}_N^*$ un grup de ordin $\phi(N) = (p-1)(q-1)$;
- Fixăm e cu $gcd(e, \phi(N)) = 1$. Atunci $(x^e)^d = x^{ed \bmod \phi(N)} = x \bmod N = (x^d)^e$
- x^d se numeste rădăcina de ordin e a lui x modulo N
- Prezumpția RSA: cunoscându-se doar N și e, este dificil să calculăm rădăcina de ordin e a unui mesaj $m \in \mathbb{Z}_N^*$

Semnatura digitală bazată pe RSA

- Gen: generează N, e, d și $pk = (N, e)$ iar $sk = d$
- $\mathrm{Sign}(sk, m)$: semnează mesajul m folosind cheia $sk = d$ astfel

$$\sigma = m^d \bmod N$$

- $\mathrm{Vrfy}(pk, m, \sigma)$: semnătura este validă dacă și numai dacă

$$m \stackrel{?}{=} \sigma^e \bmod N$$

Această variantă de semnătură nu este sigură, se cunosc mai multe atacuri pentru ea

Un atac asupra semnăturii bazate pe RSA

- scop adversar: falsificarea semnăturii mesajului $m \in \mathbb{Z}_N^*$ pentru $pk = (N, e)$
- acțiune adversar: alege $m_1, m_2 \in \mathbb{Z}_N^*$ a.i. $m = m_1 \cdot m_2 \bmod N$
- obține semnăturile σ_1 și σ_2 pentru mesajele m_1, m_2
- întoarce $\sigma = \sigma_1 \cdot \sigma_2 \bmod N$ ca semnătură validă pentru m
- aceasta este o semnătură validă pentru că

$$\sigma^e = (\sigma_1 \cdot \sigma_2)^e = (m_1^d \cdot m_2^d)^e = m_1^{ed} \cdot m_2^{ed} = m_1 \cdot m_2 = m \bmod N$$

Varianta RSA-FDH (full-domain hash)

- Gen: generează N, e, d și $pk = (N, e)$ iar $sk = d$
- $\mathrm{Sign}(sk, m)$: semnează mesajul m folosind cheia $sk = d$ astfel

$$\sigma = H(m)^d \bmod N$$

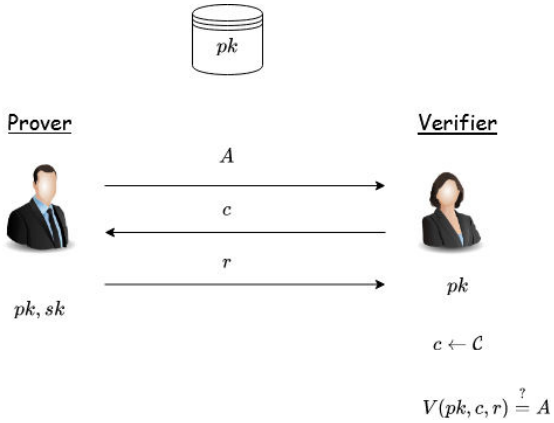
- $\mathrm{Vrfy}(pk, m, \sigma)$: semnătura este validă dacă și numai dacă

$$H(m) \stackrel{?}{=} \sigma^e \bmod N$$

- Se poate verifica ușor că atacul precedent nu funcționează: $H(m_1) \cdot H(m_1) = \sigma_1^e \cdot \sigma_2^e = (\sigma_1 \cdot \sigma_2)^e \neq H(m_1 \cdot m_2)$
  Această variantă de semnătură este sigură daca H îndeplinește două condiții:
  - H este rezistentă la coliziuni
  - $H : \{0,1\}^* \to \mathbb{Z}_N^*$

Scheme de identificare - identification schemes

- sunt protocoale interactive care permit unei părti (Prover) să își demonstreze identitatea în fața unei alte părti (Verifier)
- sunt foarte importante ca building block pentru semnături digitale (dar, în sine, au aplicabilitate limitată)
- în continuare, abordare informală

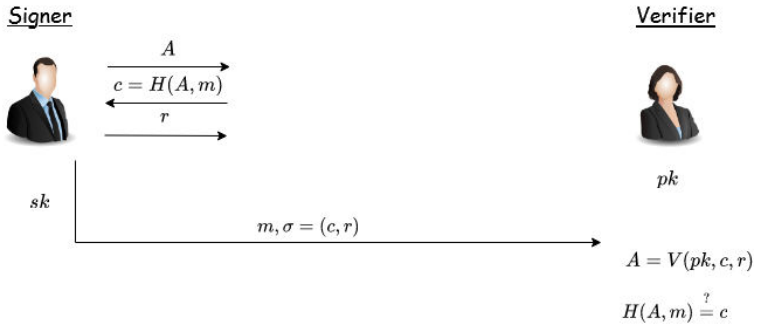


Scheme de identificare

- Securitate
  - față de adversarii *pasivi* - chiar dacă are acces la mesajele trimise în mai multe execuții ale protocolului, un adversar nu îl poate convinge pe Verifier să accepte
- Principala aplicație
  - identificarea persoanelor prezente fizic; de pildă, deschiderea unei uși securizate pe baza unei cartele de acces
  - nu este potrivită pentru autentificarea la distanță (pe internet)

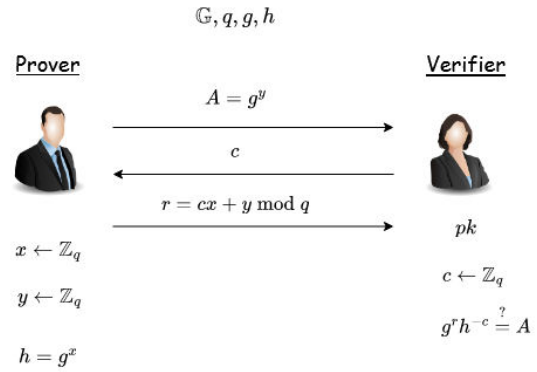
Transformarea schemelor de identificare in semnături digitale

- Pentru a semna, Prover-ul execută singur protocolul generând challenge-ul pe baza unei funcții hash - folosește *transformarea Fiat-Shamir*



Schema de identificare Schnorr

- In schema de identificare de mai jos $\mathbb{G}$ este un grup ciclic de ordin q și generator g, $sk = x$ și $pk = (\mathbb{G}, q, g, h)$ unde $h = g^x$.



- Se poate verifica ușor ca $g^r h^{-c} = g^{cx+y} h^{-c} = (g^x)^c g^y h^{-c} = g^y = A$

Schema de identificare Schnorr - securitate

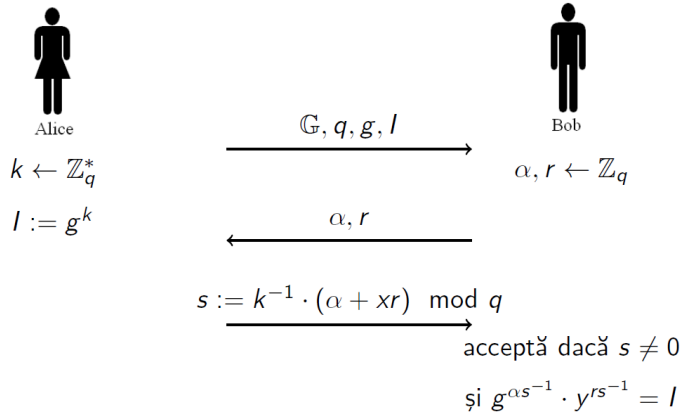
- Dacă problema logaritmului discret este dificilă, atunci schema de identificare Schnorr este sigura împotriva atacurilor pasive
- Dacă problema logaritmului discret este dificilă si H este modelata ca o funcție aleatoare, atunci semnătura Schnorr (obținută din schema de identificare Schnorr prezentată anterior, prin aplicarea transformării Fiat-Shamir) este sigură

Alte scheme de semnatură digitală

- Un alt exemplu folosit în practică este Digital Signature Algorithm (DSA) bazat pe problema logaritmului discret (a devenit standard US în 1994) dar și ECDSA (varianta DSA bazată pe curbe eliptice devenită standard în 1998), ambele fiind incluse în DSS (Digital Signature Standard).
- Atât DSA cât și ECDSA se bazează pe PLD în diferite clase de grupuri.

DSA/ECDSA

Sunt construite pe baza schemei de identificare de mai jos unde $\mathbb{G}$ este un grup ciclic de ordin q și generator g, $sk = x$ și $pk = (\mathbb{G}, q, g, y)$ unde $y = g^x$.



DSA/ECDSA

- Se poate verifica ușor că schema este corectă: $s = 0$ doar dacă $\alpha = -xr \mod q$ ceea ce se întâmplă cu probabilitate neglijabilă.
- Considerând $s \neq 0$, $s^{-1} \mod q$ există și

$$g^{\alpha s^{-1}} \cdot y^{rs^{-1}} = g^{\alpha s^{-1}} \cdot g^{xrs^{-1}} = g^{(\alpha+xr)s^{-1}} = g^{(\alpha+xr)\cdot k\cdot(\alpha+xr)^{-1}} = I$$

- Schema anterioară este sigură dacă PLD este dificilă în grupul $\mathbb{G}$.
- Schemele de semnătură DSA/ECDSA se obțin prin transformarea schemei de sus într-una non-interactivă.

DSA/ECDSA

Modificările pentru a face schema non-interactivă sunt următoarele:

- notăm $\alpha = H(m)$ unde H este o funcție hash criptografică
- $r = F(I)$ pentru o funcție specifică $F : \mathbb{G} \to \mathbb{Z}_q$
- Varianta non-interactivă a schemei este mai jos
  - Gen: generează $\mathbb{G}$ un grup ciclic de ordin q și un generator g, alege uniform $x \in \mathbb{Z}_q$ și $y = g^x$. Cheia publică este $pk = (\mathbb{G}, q, g, y)$ iar cheia secretă este $sk = x$. Se aleg și functiile $H : \{0,1\}^* \to \mathbb{Z}_q$ și $F : \mathbb{G} \to \mathbb{Z}_q$.
  - $\mathrm{Sign}(sk, m)$: alege uniform $k \in \mathbb{Z}_q^*$ și $r = F(g^k)$. Calculează $s := k^{-1} \cdot (H(m) + xr) \mod q$. Dacă $s = 0$ sau $r = 0$ se re-începe cu o nouă alegere a lui k. Semnătura rezultată este (r, s).
  - $\mathrm{Vrfy}(pk, m, (r, s))$: semnătura este validă dacă și numai dacă

$$r \stackrel{?}{=} F(g^{H(m)\cdot s^{-1}} y^{r\cdot s^{-1}})$$

DSA/ECDSA - Securitate

- Securitatea semnăturii DSA/ECDSA se bazează pe problema logaritmului discret și pe faptul că F și G sunt alese corespunzător.
- Este foarte important ca la generarea semnăturii k să fie ales aleator, și deci sa nu fie predictibil. In caz contrar, cheia secretă x se poate afla (în ecuația $s = k^{-1} \cdot (H(m) + xr) \mod q$, singura necunoscută este x).
- De asemenea, folosirea aceluiași k pentru generarea a două semnături diferite duce la găsirea cheii secrete.

Certificate și PKI

- O problemă a criptografiei cu cheie publică o reprezintă distribuirea cheilor publice;
- Se rezolvă tot cu criptografia cu cheie publică: e suficient să distribuim o singură cheie publică în mod sigur...
- Ulterior ea poate fi folosită pentru a distribui sigur oricât de multe chei publice;
- Ideea constă în folosirea unui *certificat digital* care este o semnătură care atașează unei entități o anume cheie publică;

Certificate și PKI

- De exemplu, dacă Charlie are cheia generată (pk_C, sk_C) iar Bob are cheia (pk_B, sk_B), iar Charlie cunoaște pk_B atunci el poate calcula semnătura de mai jos pe care i-o dă lui Bob:

$$\mathrm{cert}_{C\to B} = \mathrm{Sign}_{sk_C}(\text{"}\textit{Cheia lui Bob este } pk_B\text{"})$$

- Această semnătură este un *certificat* emis de Charlie pentru Bob;
- Atunci când Bob vrea să comunice cu Alice, îi trimite întâi cheia publică pk_B împreună cu certificatul $\mathrm{cert}_{C\to B}$ a cărui validitate în raport cu pk_C Alice o verifică;

Certificate și PKI

- Rămân câteva probleme: cum află Alice pk_C, cum poate fi Charlie sigur că pk_B este cheia publică a lui Bob, cum decide Alice dacă să aibă încredere în Charlie;
- Toate aceastea sunt specificate într-o *infrastructură cu chei publice* (PKI-public key infrastructure) care permite distribuirea la scară largă a cheilor publice;
- Există mai multe modele diferite de PKI, după cum vom vedea în continuare;

PKI cu o singură autoritate de certificare

- Aici există o singură autoritate de certificare (CA) în care toată lumea are încredere și care emite certificate pentru toate cheile publice;
- CA este o companie, sau agenție guvernamentală sau un departament dintr-o organizație;
- Oricine apelează la serviciile CA trebuie să obțină o copie legitimă a cheii ei publice pk_{CA};
- Cheia pk_{CA} se obține chiar prin mijloace fizice; deși inconvenient, acest pas este efectuat o singură dată;

PKI cu mai multe autorități de certificare

- Modelul cu o singură CA nu este practic;
- In modelul cu multiple CA, dacă Bob dorește să obțină un certificat pentru cheia lui publică, poate apela la oricare CA dorește, iar Alice, care primește un certificat sau mai multe, poate alege în care CA să aibă încredere;
- De exemplu, browser-ele web vin preconfigurate cu un număr de chei publice ale unor CA stabilite ca toate fiind de încredere în mod egal (în configurația default a browser-ului);
- Utilizatorul poate modifica această configurație așa încât să accepte doar certificate de la CA-uri în care el are încredere;

Delegare și lanțuri de certificate

- Charlie este un CA care emite certificate, inclusiv pentru Bob;
- Dacă pk_B este o cheie publică pentru semnătură, atunci Bob poate emite certificate pentru alte persoane; un certificat pentru Alice are forma

$$\mathrm{cert}_{B\to A} = \mathrm{Sign}_{sk_B}(\text{"}\textit{Cheia lui Alice este } pk_A\text{"})$$

- Atunci când comunică cu Dan, Alice îi trimite

$$pk_A, \mathrm{cert}_{B\to A}, pk_B, \mathrm{cert}_{C\to B}$$

- De fapt, $\mathrm{cert}_{C\to B}$ conține, în afară de pk_B și afirmația "Bob este de încredere pentru a emite certificate"; astfel, Charlie îl deleagă pe Bob să emită certificate;
- Totul se poate organiza ca o ierarhie unde există un CA "rădăcină" pe primul nivel și n CA-uri pe al doilea nivel.

Modelul "web of trust"

- Aici oricine poate emite certificate pentru orice altcineva și fiecare utilizator decide cât de multă încredere poate acorda certificatelor emise de alți utilizatori;
- De exemplu, dacă Alice are cheile publice pk_1, pk_2, pk_3 corespunzătoare lui C_1, C_2, C_3...
- ...iar Bob, care vrea să comunice cu Alice, are certificatele $\mathrm{cert}_{C_1\to B}$, $\mathrm{cert}_{C_3\to B}$ și $\mathrm{cert}_{C_4\to B}$ pe care i le trimite lui Alice;
- Alice nu are pk_4 și nu poate verifica $\mathrm{cert}_{C_4\to B}$; deci ca să accepte pk_B, Alice trebuie să decidă cât de multă încredere are în C_1 și C_3;
- Modelul e atractiv pentru că nu necesită încredere într-o autoritate centrală;

Invalidarea certificatelor

- Atunci când un angajat părăsește o companie sau își pierde cheia secretă, certificatul lui trebuie invalidat;
- Există mai multe metode de invalidare între care:
- **Expirarea.** Se poate include data de expirare ca parte a unui certificat, care trebuie verificată împreună cu validitatea semnăturii;
- **Revocarea.** CA-ul poate, în mod explicit, revoca un certificat de îndată ce acesta nu mai poate fi folosit;
- Aceasta se poate realiza prin includerea unui număr serial în certificat; la sfârșitul unei zile CA generează o listă de certificate revocate (care conține numerele seriale) pe care o distribuie sau publică.

Important de reținut!

- Semnături electronice
- Certificate digitale

Securitatea Sistemelor Informatice

- Curs 12 -
Criptografia post-cuantică

Adela Georgescu

Facultatea de Matematică și Informatică
Universitatea din București
Anul universitar 2022-2023, semestrul I

Criptografia post-cuantică

- In cadrul criptografiei de până acum am discutat despre un adversar PPT care rulează în timp polinomial pe un *calculator convențional (clasic)*. In evaluarea securității primitvelor criptografie am considerat numai *atacuri clasice*.
- Nu am avut in vedere *calculatoarele cuantice* - care se bazează pe principiile mecanicii cuantice și impactul lor asupra securității
- Algoritmii cuantici pot fi, în anumite cazuri, mult mai rapizi decât cei clasici și pot avea un impact zdrobitor asupra securității primitivelor criptografice studiate.

Criptografia post-cuantică

- D.p.d.v. teoretic, impactul calculatoarelor cuantice asupra criptografiei este recunoscut din anii 1990.
- Practic, un calculator cuantic generic, pe scară largă nu există iar costurile pentru construcția lui ar fi uriașe
- Totusi, un calculator cuantic dedicat care să atace sistemele de criptare actuale ar putea apărea în decurs de câtiva ani sau zeci de ani.
- Odată ce un astfel de calculator cuantic care să poate fi folosit în practică devine disponibil, toți algoritmii cu cheie publică folosiți în prezent dar și protocoalele asociate devin vulnerabile
- Aceasta înseamnă că toate email-urile, informațiile despre cardurile cu care facem plăți online, semnături digitale, tranzacții online, datele sensibile, informațiile clasificate ale agențiilor de securitate și cele guvernamentale vor fi în pericol

Competiția NIST pentru standardizare post-cuantică

- Ca urmare, NIST a lansat în 2017 o competiție (încă în desfășurare) pentru evaluarea și standardizarea unor scheme (de criptare, de semnatura) cu cheie publică post-cuantice - care rămân sigure în fata unor algoritmi cuantici polinomiali. `https://csrc.nist.gov/projects/post-quantum-cryptography`.
- Competitia a fost lansată în 2017 și in prima runda au fost acceptate 69 de propuneri (din cele 82 primite) https://csrc.nist.gov/Projects/post-quantum-cryptography
- La începutul lui 2019, au fost aleși 26 de candidați pentru runda a 2-a.
- Iulie 2020 - anuntați pentru runda a 3-a: 7 finaliști și 8 candidati alternativi

Competiția NIST pentru standardizare post-cuantică

- Iulie 2022 - NIST anunță primul grup de 4 finaliști
  - **criptare**: CRYSTALS - Kyber - pentru chei mici de criptare și operații rapide
  - **semnături digitale**
    - CRYSTALS-Dilithium
    - FALCON
    - SPHINCS+
- Dilithium și Falcon sunt foarte eficienți, cel din urmă fiind recomandat atunci când sunt necesare semnături mai mici decât cele oferite de Dilithium
- Primii 3 algoritmi se bazează pe probleme matematice de latici, iar SPHINCS+ folosește funcții hash
- Procesul de standardizare se va finaliza în anul 2024
- Alți 4 algoritmi sunt considerați pentru standardizare

Criptografia post-cuantică vs. criptografia cuantică

Criptografia cuantică

- implementări folosind calculatoare cuantice, fenomene mecanice cuantice și canale de comunicare cuantice
- dificil de implementat la scara largă
- în unele cazuri este sigură necondiționat (nu se bazează pe ipoteze matematice)

Criptografia post-cuantică

- implementări folosind calculatoare clasice
- este sigură chiar și în fața unui adversar care are acces la un calculator cuantic
- se bazează pe probleme matematice dificile computațional chiar și pentru algoritmii cuantici

Criptografia simetrică post-cuantică

- Impactul calculatoarelor cuantice asupra criptografiei simetrice este minor; ilustram pe scurt
- Considerăm următoarea problema abstractă:
  - Se dă: funcție $f : D \to \{0,1\}$ cu acces de tip oracol (funcția poate fi interogată pe orice input și se primește output-ul corespunzător)
  - Se cere: să se găsească x a.î. $f(x) = 1$.
- Dacă există un singur x cu $f(x) = 1$ atunci orice algoritm clasic necesită $O(\|D\|)$ evaluări ale funcției f - corespunde unui atac prin forță brută
- 1996 - algoritmul cuantic al lui Grover: găsește x folosind $O(\|D^{1/2}\|)$ evaluări ale funcției f. Algorimtul este optim, și nu poate fi îmbunătățit

Criptografia simetrică post-cuantică - sisteme bloc

- Trecem in revista impactul algoritmului asupra sistemelor de criptare simetrice
- Considerăm cazul unui sistem de criptare bloc $F : \{0,1\}^n \times \{0,1\}^l \to \{0,1\}^l$ cu cheia k pe n biți pentru care cel mai bun atac (de găsire a cheii) este forța brută.
- Un astfel de atac clasic necesită timp 2^n
- Pentru securitate, alegem cheia k de lungime n biti așa încât timpul pentru atac 2^n să nu fie practic
- Algoritmul lui Grover însă permite unui atacator să găsească cheia în timp $2^{n/2}$.
- Pentru același nivel de securitate (precum în cazul clasic), alegem cheia k de lungime **dublă** fața de cazul clasic.

Criptografia simetrică post-cuantică - funcții hash

- Considerăm problema găsirii de coliziuni pentru o funcție hash $H : \{0,1\}^m \to \{0,1\}^n$ cu $m > n$.
- In cazul **clasic**, am văzut că "atacul nașterilor" necesită $O(2^{n/2})$ evaluări ale funcției H.
- Aceasta înseamnă că pentru a asigura rezistența la coliziuni fața de un atac în timp 2^t, trebuie să alegem funcții hash cu output-ul pe $2t$ biți.
- Insă în cazul **cuantic**, un atac pentru găsirea coliziunilor necesită $O(2^{n/3})$ evaluări ale funcției H.
- Deci, pentru a asigura rezistența la coliziuni față de un atac în timp 2^t, trebuie să alegem funcții hash cu output-ul pe $3t$ biți.

Algoritmul lui Shor și impactul lui asupra criptografiei asimetrice

- Până acum am văzut algoritmi cuantici care oferă o îmbunătățire de *ordin polinomial* în comparație cu cei mai buni algoritmi clasici pentru aceeași problemă.
- Aceștia impun doar creșterea dimensiunilor cheilor fără a necesita alte schimbari majore
- In continuare vom vedea un algoritm care oferă o îmbunătățire de *ordin exponențial* - **algoritmi cuantici polinomiali pentru problema factorizarii și problema logaritmului discret**

Algoritmul lui Shor și impactul lui asupra criptografiei asimetrice

- Incepem cu o problemă abstractă:
  - Se dă: o funcție $f : \mathbb{G} \to R$, cu $\mathbb{G}$ un grup comutativ.
  - Presupunem f *periodică*: există $\alpha \in \mathbb{G}$ - *perioadă*- cu $f(x) = f(x + \alpha)$
  - Se cere: găsirea unei perioade având acces de tip oracol la funcția f
- Nu se cunosc algoritmi clasici eficienți pentru rezolvarea acestei probleme.

Algoritmul lui Shor și impactul lui asupra criptografiei asimetrice

- 1994 - Shor - rezultat uimitor: *algoritm cuantic polinomial* care rezolvă problema pentru anumite grupuri $\mathbb{G}$.
- Este un instrument puternic care poate fi folosit pentru a factoriza și a calcula logaritmi discreti.
- Trebuie doar aleasă cu grijă funcția a cărei perioadă ne dă soluția pe care o căutăm

Algoritmul lui Shor și impactul lui asupra factorizarii

- Considerăm problema factorizarii: fie N un produs de două numere prime. Pentru orice $x \in \mathbb{Z}_N^*$, definim funcția $f_{x,N} : \mathbb{Z} \to \mathbb{Z}_N^*$

$$f_{x,N}(r) = x^r \bmod N$$

- Principala observație este că funcția are perioada $\phi(N)$ deoarece

$$f_{x,N}(r+\phi(N)) = x^{r+\phi(N)} \bmod N = x^r \cdot x^{\phi(N)} \bmod N = x^r \bmod N$$

- Pentru orice x ales de noi, putem calcula, in timp polinomial, cu algoritmul lui Shor, o perioadă a funcției $f_{x,N}$ adică un $k \neq 0$ cu $x^k = 1 \bmod N$. Aceasta imediat permite factorizarea lui N în timp polinomial.

Algoritmul lui Shor și impactul lui asupra criptografiei asimetrice

- Algoritmul lui Shor poate fi folosit și pentru rezolvarea problemei logaritmului discret în timp polinomial
- Având în vedere că toate sistemele de criptare cu cheie publică se bazează pe problema factorizării sau problema logaritmului discret, concluzionăm că

Toate sistemele de criptare cu cheie publică prezentate la curs pot fi atacate în timp polinomial cu ajutorul unui calculator cuantic

Sisteme de criptare cu cheie publică post-cuantice

- Problema factorizării și problema logaritmului discret devin "ușoare" pentru un calculator cuantic.
- Avem nevoie de probleme matematice dificile computațional chiar și pentru calculatoarele cuantice, dar care rulează pe calculatoare clasice
- Diferenta față de cazul clasic este că problemele considerate pentru criptografia post-cuantică sunt mai recente și nu au fost studiate la fel de mult ca problema factorizării sau problema logaritmului discret
- In continuare vom prezenta o problemă care a primit multa atenție și care este considerată dificilă chiar și pentru calculatoarele cuantice. Aratam apoi cum se poate construi un sistem de criptare cu cheie publica bazat pe dificultatea acelei probleme.

Problema LWE - Learning With Errors

- A fost introdusa in 2005 de Oded Regev
- Preliminarii:
  - q număr prim. Vom nota cu $\mathbb{Z}_q$ mulțimea $\{-\lfloor (q-1)/2 \rfloor, ..., 0, ..., \lfloor q/2 \rfloor\}$ (spre deosebire de $\{0, ..., q\}$) unde $\lfloor x \rfloor$ este cel mai mare intreg mai mic sau egal cu x.
  - spunem că un element al lui $\mathbb{Z}_q$ este "mic" dacă este "aproape" de 0.
- Problema cere găsirea lui $\mathbf{s} \in \mathbb{Z}_q^n$ fiind dată o secvență de ecuații liniare "aproximative" în $\mathbf{s}$.
- Exemplu:

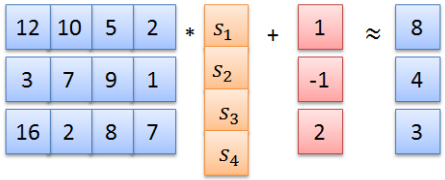
$$12s_1 + 10s_2 + 5s_3 + 2s_4 \approx 8 \bmod 17$$
$$3s_1 + 7s_2 + 9s_3 + s_4 \approx 4 \bmod 17$$
$$16s_1 + 2s_2 + 8s_3 + 7s_4 \approx 3 \bmod 17$$

Problema LWE - Learning With Errors

- Exemplul

$$12s_1 + 10s_2 + 5s_3 + 2s_4 + 1 = 8 \bmod 17$$
$$3s_1 + 7s_2 + 9s_3 + s_4 - 1 = 4 \bmod 17$$
$$16s_1 + 2s_2 + 8s_3 + 7s_4 + 2 = 3 \bmod 17$$

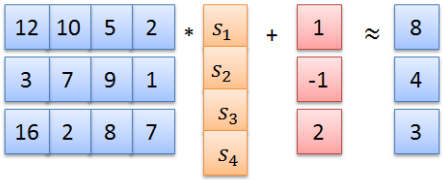
sub forma matriciala



sau, notand matricile corespunzator (unde $\mathbf{s} = (s_1, s_2, s_3)$), ecuația matricială devine

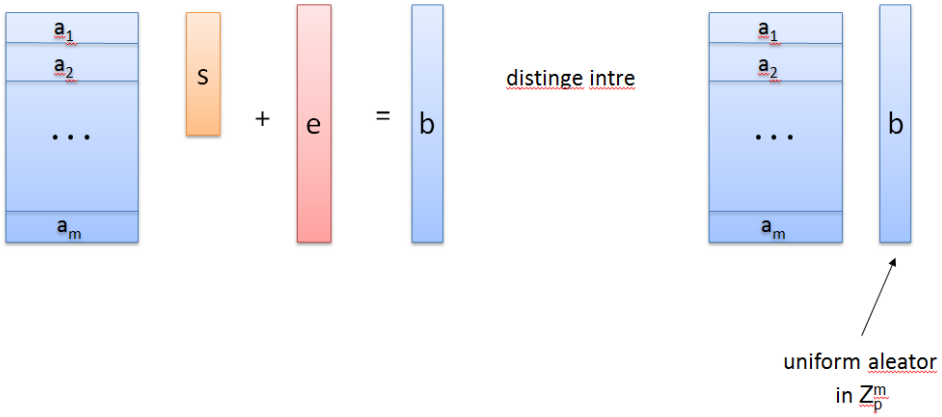
$$\mathbf{As} + \mathbf{e} = \mathbf{b} \bmod q$$

Problema LWE - Learning With Errors



- vectorul $\mathbf{e} = (1, -1, 2)$ este format din elemente "mici" din $\mathbb{Z}$ numite *noise* sau *error*.
- In lipsa lui **e**, ecuația $\mathbf{As} = \mathbf{b}$ devine usor de rezolvat cu tehnici clasice de algebra liniară
- Cand matricea **A** are suficient de multe linii și parametrii sunt aleși corespunzător, problema devine dificilă.

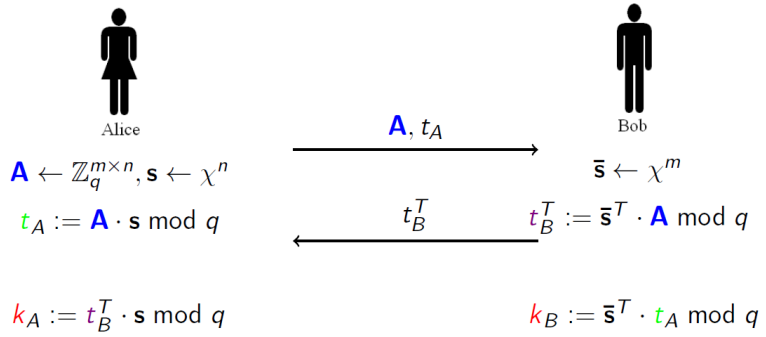
Problema decizională LWE



- Problema decizională cere să se distingă între un **b** generat ca mai sus (stânga) și un **b** generat uniform aleator în $\mathbb{Z}_q^m$.

Sistem de criptare bazat pe LWE

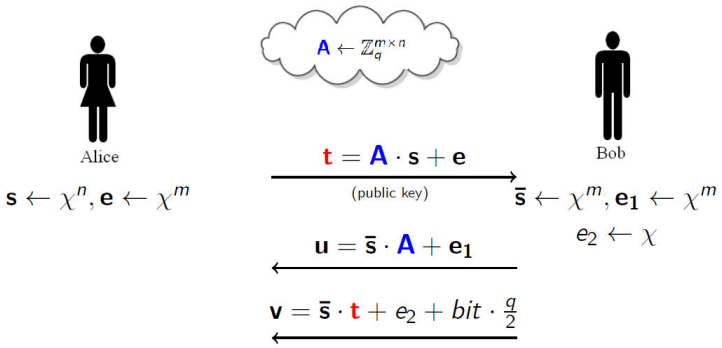
Descriem mai întâi un schimb de chei (nesigur) care poate fi văzut ca o versiune algebric liniară a schimbului de chei Diffie-Hellman. Fixează parametrii $n, q, \chi, m > n$



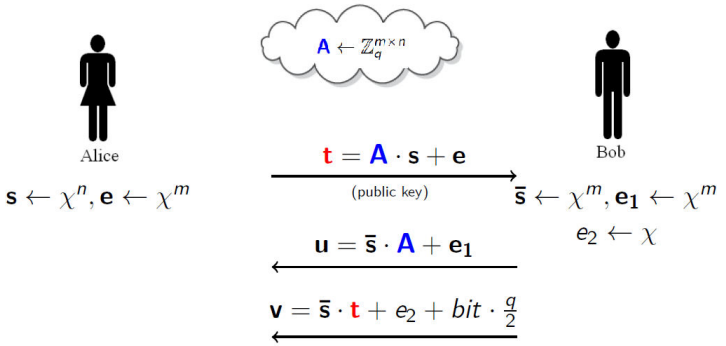
- Verificăm ușor că Alice și Bob partajează aceeași cheie $k_A := t_B^T \cdot \mathbf{s} = \bar{\mathbf{s}}^T \cdot \mathbf{A} \cdot \mathbf{s} = \bar{\mathbf{s}}^T \cdot t_A = k_B$

Sistem de criptare bazat pe LWE

- Protocolul de mai sus nu este sigur pentru că un atacator poate calcula $\bar{\mathbf{s}}$ sau $\mathbf{s}$ cu noțiuni de algebră liniară și poate afla și cheia.
- Insă el poate fi transformat într-un protocol sigur și adaptat ca un sistem de criptare adăugând "noise", sub ipoteza problemei decizionale LWE.



Sistem de criptare bazat pe LWE



- Schema de mai sus criptează un bit iar decriptarea se face calculând $x = \mathbf{v} - \mathbf{u} \cdot \mathbf{s}$.
- Rezultatul va fi 1 dacă x este mai aproape de $\frac{q}{2}$ decât de 0.
- Apropierea lui x de $\frac{q}{2}$ este verificată calculând valoarea absolută a lui $x - \frac{q}{2}$ mod q

Sistem de criptare bazat pe LWE - securitate

- Decriptarea funcționeaza corect atâta timp cât $|\bar{\mathbf{s}} \cdot \mathbf{e} + e_2 - \mathbf{e_1} \cdot \mathbf{s}| < (q-1)/4$
- Această condiție este îndeplinită dacă distribuția χ a valorilor $\mathbf{s}, \bar{\mathbf{s}}, \mathbf{e}, \mathbf{e_1}, e_2$ este aleasă corect și produce numere întregi suficient de mici.
- Sistemul de criptare este CPA-sigur (chiar și pentru adversari cuantici) dacă problema decizională LWE este dificilă.

Exerciții

Fie El Gamal cu $pk = (g, h = g^a)$ și $sk = (g, a)$ în $\mathbb{G}$.

- **Enc**: dată o cheie publică $(\mathbb{G}, q, g, h)$ și un mesaj $m \in \mathbb{G}$, alege $y \leftarrow^R \mathbb{Z}_q$ și întoarce $c = (c_1, c_2) = (g^y, m \cdot h^y)$;
- **Dec**: dată o cheie secretă $(\mathbb{G}, q, g, a)$ și un mesaj criptat $c = (c_1, c_2)$, întoarce $m = c_2 \cdot c_1^{-a}$.

Vrem să distribuim cheia secretă la două persoane așa încât numai cele două persoane împreună pot decripta. O modalitate simplă de a rezolva această problemă este să alegem două numere aleatoare $a_1, a_2 \in \mathbb{Z}_n$ așa încât $a_1 + a_2 = a$. O persoană primește a_1 iar cealaltă primește a_2. Pentru decriptarea (c_1, c_2), trimitem c_1 ambelor persoane.

Ce valori trebuie să calculeze cele două persoane și să ne trimită înapoi așa încât să putem decripta textul criptat trimis?

Solution

Persoana 1 trimite $u_1 = c_1^{a_1}$ iar persoana 2 trimite $u_2 = c_1^{a_2}$. Produsul $u_1 \cdot u_2 = c_1^{a_1+a_2} = c_1^a$ împreună cu c_2 poate fi folosit la decriptare.

Se consideră $H : \{0,1\}^* \to \{0,1\}^n$ o funcție hash rezistentă la a doua preimagine și rezistentă la coliziuni. Se definește o funcție $H^* : \{0,1\}^* \to \{0,1\}^{n+1}$ astfel:

$$H'(x) = \begin{cases} x||1 & \text{dacă } x \in \{0,1\}^n \\ H(x)||0 & \text{altfel} \end{cases}$$

Argumentați că H' este rezistentă la coliziuni.

Solution

Fie $H'(x_1) = H'(x_2)$ cu $x_1 \neq x_2$. Dacă $H'(x_1) = x||1$ rezultă $x_1 = x_2$, contradicție. Dacă $H'(x_1) = H(x_1)||0 = H(x_2)||0$ atunci se determină o coliziune pentru H, contradicție.

Fie (Mac, Vrfy) un MAC sigur definit peste (K,M,T) unde $M = \{0,1\}^n$ și $T = \{0,1\}^{128}$. Este MAC-ul de mai jos sigur? Argumentați răspunsul.

$$Mac'(k,m) = Mac(k,m)$$

$$Vrfy'(k,m,t) = \begin{cases} Vrfy(k,m,t), & \text{dacă } m \neq 0^n \\ 1, & \text{altfel} \end{cases}$$

Solution

MAC-ul nu este sigur pentru ca un adversar poate sa intoarca perechea validă $(0^n, 0^s)$.

Securitatea Sistemelor Informatice

- Curs 13 - TLS

Adela Georgescu

Facultatea de Matematică și Informatică
Universitatea din București
Anul universitar 2022-2023, semestrul I

TLS - Transport Layer Security

- Este un protocol folosit de browser-ul web de fiecare data cand ne conectam la un browser folosind `https`
- primele versiuni se numeau SSL - Secure Sockets Layer - dezvoltat de Netscape (1995) - SSL 3.0, cea mai cunoscută versiune
- TLS 1.0 apare în 1999, TLS 1.1 în 2006, TLS 1.2 în 2008 și versiunea actuală, sigura și eficientă TLS 1.3 în 2018
- folosirea lui SSL 3.0, TLS 1.0 și TLS 1.1 trebuie evitată, toate trei prezintă probleme de securitate
- se recomandă folosirea minim a lui TLS 1.2

TLS - Transport Layer Security

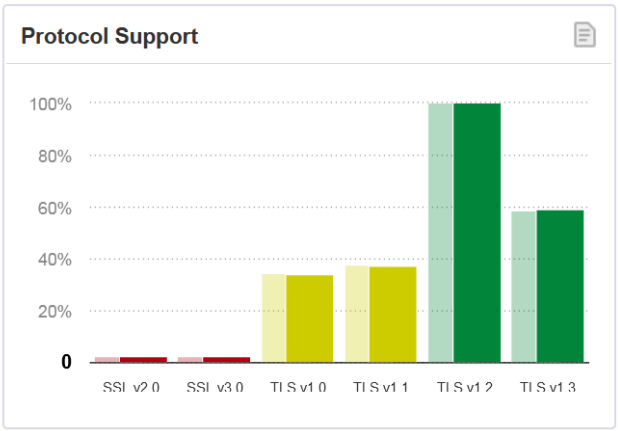
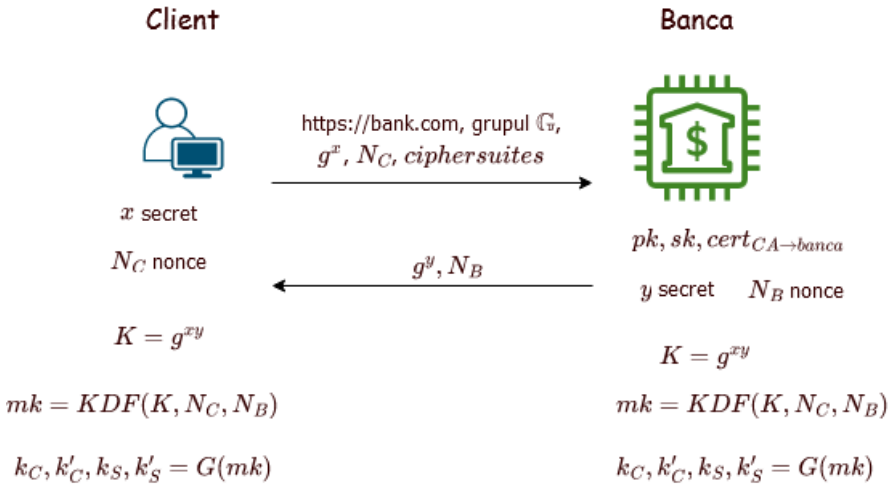


Figure: Acoperirea versiunilor de TLS conform SSL Pulse (decembrie 2022)

TLS - Transport Layer Security

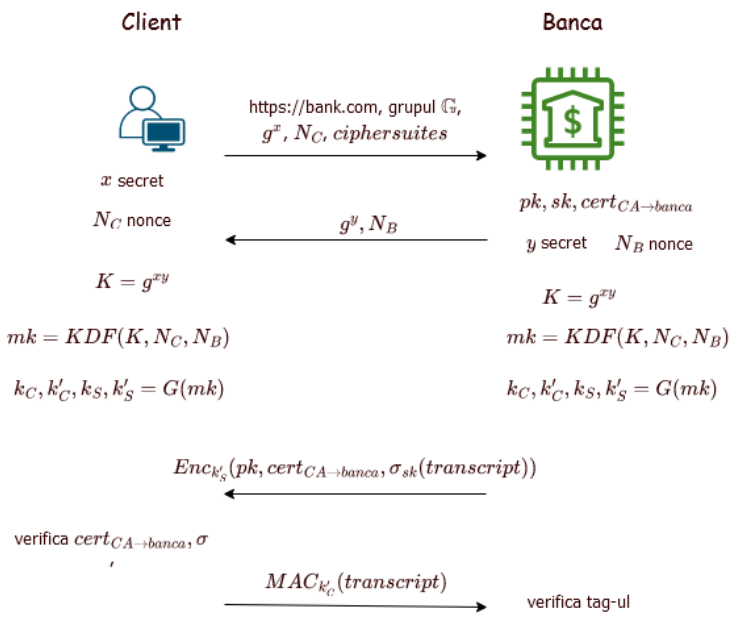
- Protocolul TLS permite unui client (de ex. browser web) și unui server (de ex. website) să se pună de acord asupra unui set de chei pe care să le folosească ulterior pentru comunicare criptată si autentificare
- Protocolul TLS constă din 2 părți:
  1. *protocolul handshake* - realizează schimbul de chei care stabilește un set de chei comune
  2. *protocolul record-layer* - folosește cheile stabilite pentru criptare/autentificare ulterioară
- In continuare vom prezenta protocolul handshake din versiunea curentă TLS 1.3

Handshake protocol (TLS 1.3)...



- KDF = key derivation function - algoritm criptografic pentru derivarea de chei, pe baza unui PRF
- G = generator de numere pseudo-aleatoare (PRG)

...Handshake protocol...



...Handshake protocol

- *ciphersuites* - colecție de algoritmi criptografici
- TLS 1.3 suportă 5 ciphersuites:
  - TLS_AES_128_GCM_SHA256
  - TLS_AES_256_GCM_SHA384
  - TLS_CHACHA20_POLY1305_SHA256
  - TLS_AES_128_CCM_SHA256
  - TLS_AES_128_CCM_8_SHA256

- *transcript* - reprezintă mesajele trimise în cadrul protocolului până la momentul curent
- cheile k'_S și k'_C sunt folosite doar în handshake

TLS 1.3

- clientul și serverul, partajează, la finalul protocolului, cheile k_S și k_C, pe care le folosesc ulterior pentru criptare și autentificare
- în plus, clientul are garanția că la final a partajat cheile cu server-ul legitim și că acestea nu au fost interceptate sau modificate de o terță parte de ce?
- în handshake-ul din TLS 1.2, clientul și server-ul foloseau o schemă de criptare cu cheie publică în locul schimbului de chei Diffie-Hellman
- clientul doar alegea o cheie K pe care o trimitea serverului criptată cu cheia lui publică

TLS 1.3

- aceasta a fost eliminată în TLS 1.3 pentru că nu asigură proprietatea de *forward secrecy* - care presupune că, în cazul compromiterii server-ului, cheile de sesiune anterioare nu sunt compromise
- în TLS 1.3, în ceea ce priveste server-ul, cheia K este calculata pe baza secretului y in fiecare sesiune, secret care este unul nou de fiecare data si care nu trebuie mentinut intre sesiuni; deci compomiterea server-ului (aflarea cheii lui secrete) nu duce la aflarea cheii K
- în TLS 1.2, cheia secretă K ii este trimisa server-ului criptata cu cheia lui publica; compromiterea cheii secrete server-ului duce la compromiterea cheilor de sesiune din toate sesiunile
- în protocolul *record*, clientul și server-ul comunică în mod confidențial și autentificat folosind o schemă de criptare autentificată

TLS 1.3 vs. TLS 1.2

- număr redus de runde

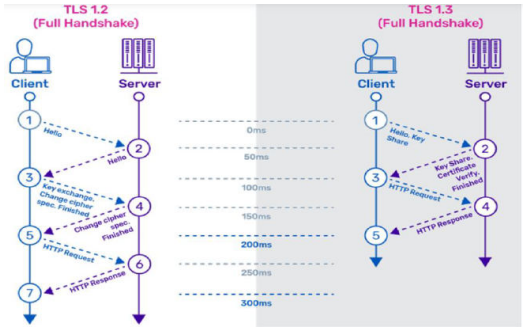


Figure: Sursa: www.a10networks.com

- eliminarea algoritmilor criptografici vulnerabili precum SHA-1, RC4, DES, 3DES, AES-CBC, MD5
- 0-RTT (zero round-trip) – reluarea unei sesiuni mai vechi cu un website e mult mai rapida

Securitatea Sistemelor Informatice

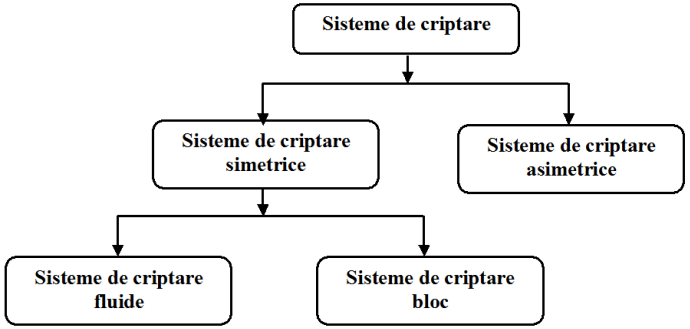
- Curs 14 -
Mai multe despre criptografie

Adela Georgescu

Facultatea de Matematică și Informatică
Universitatea din București

Primitive criptografice studiate

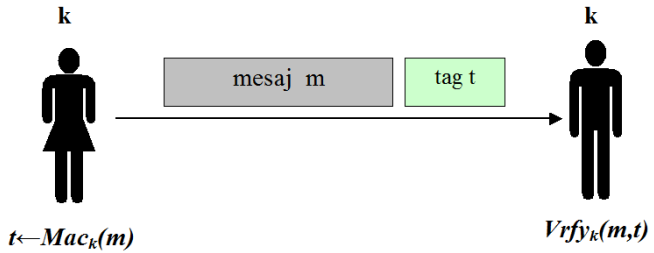
- Am studiat în timpul cursului **sisteme de criptare**:



- Acestea au rolul de a asigura **confidențialitatea**.

Primitive criptografice studiate

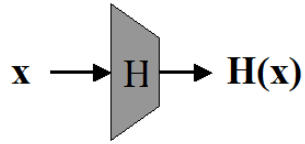
- Am studiat în timpul cursului construcțiile **MAC**, dar și **semnăturile digitale**:



- Acestea au rolul de a asigura **integritatea**.

Primitive criptografice studiate

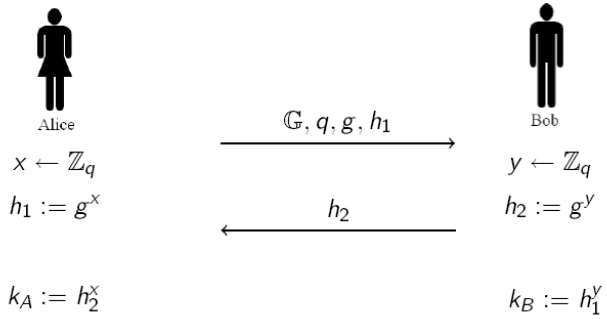
- Am studiat în timpul cursului funcții **hash**:



- Acestea asigură **integritatea** și **autentificarea** datelor prin utilizare în MAC-uri, semnături digitale...

Primitive criptografice studiate

- Am studiat în timpul cursului, protocoale de **schimb de chei** (Diffie-Hellman):



- Acestea asigură **stabilirea unei chei comune**, utilizată ulterior în scopuri criptografice (ex. criptare).

Alte primitive criptografice

- Am studiat sisteme de criptare care asigură confidențialitatea între 2 participanți;
- Există însă și sisteme de criptare de tip **broadcast** (**broadcast encryption**);
- Acestea permit comunicarea criptată (unidirecțională) peste un canal de tip *broadcast* (către toți participanții) a.î. numai participanții autorizați să poată decripta;
- Exemple de utilizare: transmisiuni TV criptate;
- Noțiuni similare:
  - **threshold encryption**: pentru decriptare este necesar să coopereze un număr de participanți care să depășească un anumit prag.

Alte primitive criptografice

- Am studiat construcțiile MAC care asigură integritatea în criptografia simetrică;
- Dar și **semnăturile digitale**, care atestă în plus și originea mesajului.

Alte primitive criptografice

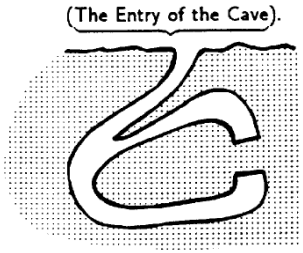
- Am studiat protocolul de schimb Diffie-Hellman care stabilește o cheie comună între 2 participanți;
- Există însă și **protocoale de stabilire a cheilor de grup**;
- Acestea permit stabilirea unei chei comune între mai mulți participanți;
- Exemple de utilizare: comunicație criptată, (video-) conferințe, acces la resurse ...

Alte primitive criptografice

- **Schemele de partajare a secretelor** permit partajarea unui secret în mai multe componente distribuite unor participanți astfel încât numai mulțimile *autorizate* de participanți să poată reconstitui secretul;
- Exemple de utilizare: controlul accesului, stocarea fișierelor în cloud, ...
- Alte primitive criptografice la nivel de grup: **multiparty computation**, **protocoale de vot elecronic** ...

Alte primitive criptografice

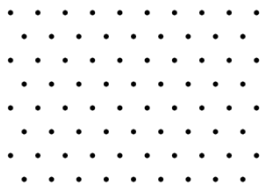
- Protocoale de tip **zero-knowledge** permit unei entități (*prover*) să demonstreze cunoașterea unui secret unei alte entități (*verifier*) fara a dezvalui nici un fel de informatie despre secret;



[J.J.Quisquater, L.C.Guillou, T.A.Berson,
How to Explain Zero-Knowledge Protocols to Your Children]

Mai mult despre criptografie

- Am studiat criptografia bazată pe teoria numerelor și criptografia bazată pe curbe eliptice;
- Dar există și alte tipuri de criptografie, precum **criptografia bazată pe latici**:



$$\{\sum_{i=1}^{n} a_i v_i | a_i \in \mathbb{Z}, v_i \text{ bază}\}$$

- Probleme dificile: SVP (Shortest Vector Problem), CVP (Closest Vector Problem), ...

Mai mult despre criptografie

- **Criptografia cuantică**: