

**Proprietati securitate asigura de criptografie:**

- Confidentialitatea (mesajelor) - adversarul nu vede sau nu poate obtine mesajul  $M \Rightarrow$  Criptare
- Integritatea (mesajelor) – Alice (sau Bob) trebuie sa isi dea seama daca mesajul primit a fost modificat – asigurata de MAC sau semnaturi digitale
- Autentificarea (expeditorului si a mesajului) – Bob trebuie sa poata verifica ca mesajul provine de la Alice
- Ne-repudierea - Alice nu poate nega ca a trimis mesajul lui Bob
- !! Integritate + Confidentialitate  $\Rightarrow$  Criptare autentificata

**Criptare simetrica (cu cheie secreta)**

- $K$  = spatiul cheilor
- $M$  = spatiul mesajelor clare
- $C$  = spatiul textelor criptate
- (Gen,Enc,Dec) – sistem de criptare simetric unde Gen – algoritm probabilistic de generare a cheilor care intoarce o cheie  $k$  conform unei distributii
- $Pr[K=k]$  – probabilitatea cheii generate de Gen de a lua valoarea  $k$

**Scenarii de atac:**

- Ciphertext-only attack – atacatorul stie doar textul criptat, poate incerca un brute-force prin care se parcurg toate cheile pana se gaseste cea corecta
- Known-plaintext attack – atacatorul stie una sau mai multe perechi (text clar, text criptat)
- Chosen-Plaintext attack – atacatorul poate obtine criptarea unor texte clare alese de el !! atacator activ
- Chosen-ciphertext attack – atacatorul are posibilitatea sa obtina decriptarea unor texte criptate alese de el !! atacator activ

**Principiul cheilor suificienta** – o schema sigura de criptare  $\Rightarrow$  nu poate fi sparta cu brute-force

**One Time Pad (OTP)**

- Avantaj – criptare si decriptare rapide
- Dezavantaj – cheia foarte lunga (la fel de lunga precum textul clar)
- Acelasi text criptat poate sa provina din orice text clar cu o cheie potrivita
- Daca adversarul nu stie decat textul criptat, atunci nu stie nimic despre textul clar
- Daca OTP este folosit de mai multe ori si adversarul obtine  $C = M \text{ xor } K$  si  $C' = M' \text{ xor } K \Rightarrow$  el poate obtine  $M \text{ xor } M' \Rightarrow$  nu mai este sigur
- Securitatea perfecta a OTP :
  - o Cheia trebuie sa fie la fel de lunga precum mesajul
  - o Inconveniente practice (stocare,transmisie)
  - o Cheia trebuie sa fie folosita o singura data

**Teorema** – Nu exista nici o schema de criptare (Enc,Dec) perfect sigura in care mesajele au lungimea  $n$  biti iar cheile au lungimea (cel mult)  $n-1$  biti

**Criptografia istorica (abordare ad-hoc) – 1980 – criptografia moderna (abordare riguroasa)**

**Securitate perfecta (Shannon 1949)** – cel mai puternic adversar nu poate deduce nimic despre textul clar dat fiind textul criptat

**OTP este perfect sigura**

**Scheme de criptare sigure chiar si in fata unui adversar cu putere computationala nelimitata**  $\rightarrow$  informational-teoretic sigure

**Majoritatea schemelor moderne**  $\rightarrow$  securitatea computationala care este mai slaba decat cea informational-teoretica (poate fi sparta cu putere mare a adversarului, se face un compromis de securitate pt a obtine constructii practice)

**Pseudoaleatorismul**

- Un sir pseudoaleator "arata" similar unui sir uniform aleator din punct de vedere al oricarui algoritm polinomial
- Altfel spus: un algoritm polinomial nu poate face diferenta intre o secventa perfect aleatoare si una pseudoaleatoare (decat cu probabilitate neglijabila)
- Sau: o distributie a secventelor de lungime  $l$  este pseudoaleatoare daca este nedistinctibila de distributia uniforma a secventelor de lungime  $l$

**PRG (Generator de numere aleatoare)**

- Acesta este un algoritm determinist care primeste o "samanta" relativ scurta  $s$  (seed) si genereaza o secventa pseudoaleatoare de biti
- $l(n) \geq n$  (factor de expansiune)
- Distributia output-ului unui PRG este departe de a fi uniforma
- Seedul unui PRG analog cheii unui sistem de criptare (ales uniform si mentinut secret, suficient de lunga ca sa nu permita brute-force)

**Sistem de criptare bazat pe PRG**  $\Rightarrow$  Daca  $G$  este PRG, atunci sistemul definit anterior este un sistem de criptare simetric de lungime fixa computational sigur pentru un atacator pasiv care poate intercepta un mesaj.

- Criptarea bazata pe PRG se obtine din OTP prin inlocuirea pad cu  $G(k)$
- Daca  $G$  este PRG, atunci pad si  $G(k)$  sunt indistinctibile pentru orice  $A$  adversar PPT
- In concluzie, OTP si sistemul de criptare bazat pe PRG sunt indistinctibile pentru  $A$

**Teorema** - O schema de criptare (Enc, Dec) unde functia Enc este determinista nu are proprietatea de securitate la interceptare multipla

Pentru a obtine securitate la interceptare multipla, avem nevoie de o schema de criptare probabilista, asa incat la criptari succesive ale aceluasi mesaj sa obtinem texte criptate diferite

**Definitie** - O schema de criptare  $\pi = (\text{Enc}, \text{Dec})$  este CPA-sigura daca pentru orice adversar PPT A exista o functie neglijabila  $\text{negl}$  asa incat  $\Pr[\text{Priv}_{A,n}^{\text{CPA}}(n) = 1] \leq \frac{1}{2} + \text{negl}(n)$ . => Un adversar nu poate determina care text clar a fost criptat cu o probabilitate semnificativ mai mare decat daca ar fi ghicit (in sens aleator, dat cu banul), chiar daca are acces la oracolul de criptare.

**CPA-Sigur** are mereu caracter de indistinctibilitate. UN sistem de criptare determinist NU poate fi CPA-Sigur

**PRG conditionat** , dpdv teoretic, poate fi consturit pe baza functiilor one-way. Presumtie: PRG exista.

- In practica PRG-urile sunt instantiate cu sisteme de criptare fluide.
- sistemele fluide produc bitii de output (pseudo-aleatori) gradual si la cerere, fiind mai eficiente si flexibile
- Faza 1: se genereaza o secventa pseudoaleatoare de biti, folosind un generator de numere pseudoaleatoare (PRG)
- Faza 2: secventa obtinuta se XOR-eaza cu mesajul clar
- utilizarea unui sistem fluid in forma prezentata pentru criptarea

**Moduri de utilizare a unui sistem de criptare fluid:**

- **Modul sincronizat**
- mesajele sunt criptate in mod succesiv (participantii trebuie sa stie care parti au fost deja folosite)
- necesita pastrarea starii
- mesajele succesive pot fi percepute ca un singur mesaj clar lung, obtinut prin concatenarea mesajelor succesive
- se preteaza unei singure sesiuni de comunicatii
- **Modul nesincronizat**
- mesajele sunt criptate in mod independent
- NU necesita pastrarea starii
- valorile IV1, IV2, . . . sunt alese uniform aleator pentru fiecare mesaj transmis
- valorile IV1, IV2, . . . (dar si IV in modul sincronizat) fac parte din mesajul criptat (sunt necesare pentru decriptare)

**Linear-Feedback Shift Registers (LFSR)**

- sunt foarte eficiente in implementari hardware
- au proprietati statistice bune dar totusi sunt predictibile, deci nu sunt PRG-uri sigure din punct de vedere criptografic
- starea LFSR consta din n biti (continutul registrilor la un moment dat)
- exista cel mult  $2^n$  stari posibile pana cand output-ul LFSR-ului se repeta
- cunoscand cel mult  $2^n$  biti de la iesire, un atacator poate afla starea initiala si coeficientii de feedback

**RC4 este un sistem de criptare fluid pe octeti:**

- 2 faze: 1.initializare: determina starea interna, fara sa produca chei fluide; 2.generare de chei fluide: modifica starea interna si genereaza un octet (cheia fluida) care se XOR-eaza cu m pentru a obtine c
- Starea interna: un tablou S de 256 octeti: S[0], . . . , S[255]; 2 indici i si j;
- Toate operatiile se efectueaza pe octeti (i.e. (mod 256)).

**Vulnerabilitati LFSR**

- LFSR-urile sunt liniare iar liniaritatea induce vulnerabilitati (sistemele liniare de ecuatii permit aflarea informatiilor sensibile)
- Insa combinatiile de mai multe LFSR-uri pot produce sisteme de criptare sigure

**Sisteme de criptare bloc (cate n biti simultan)**

- criptarea unui bit din textul clar este dependenta de bitii din textul clar care apartin aceluia si bloc
- necesitati computationale mai avansate
- utilizare: internet (SISTEME FLUIDE utilizare: telefoane mobile, dispozitive incorporate, PDA)
- par sa fie mai sigure, prezinta incredere mai mare
- PRP (permutare pseudoaleatoare) sunt necesare pentru constructia sistemelor bloc, def: Acesta este o functie determinista si bijectiva care pentru o cheie fixata produce la iesire o permutare a intrarii indistinctibila fata de o permutare aleatoare

**Teoremă (Luby-Rackoff 5)**

*Dacă  $F : \mathcal{K} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$  este PRF, se poate construi  $F' : \mathcal{K} \times \{0, 1\}^{2n} \rightarrow \{0, 1\}^{2n}$  PRP.*

- Lungime mesaj clar < dimensiune bloc => se completeaza cu biti 1 0 ... 0
- Lungime mesaj clar > lungime bloc => Se utilizeaza un mod de operare: ECB, CBC, OFB, CTR

**ECB (Electronic Code Book)**

- Paralelizabil, determinist (nesigur)
- Un adversar pasiv detecteaza repetarea unui bloc de text clar pentru ca se repeta blocul criptat corespunzator;
- Fk inversabil
- Modul ECB NU trebuie utilizat in practica!

**CBC (Cipher Block Chaining)**

- IV este ales in mod aleator la criptare
- IV se transmite in clar pentru ca este necesar la decriptare
- Fk inversabil
- Este secvential, un dezavantaj major daca se poate utiliza procesarea paralela

**OFB (Output FeedBack)**

- Genereaza o secventa pseudoaleatoare care se XOR-eaza mesajului clar
- Fk nu neaparat inversabil
- IV este ales in mod aleator la criptare
- IV se transmite in clar pentru ca este necesar la decriptare
- Este secvential, insa secventa pseudoaleatoare poate fi pre-procesata anterior decriptarii

- CTR (Counter)**
- Genereaza o secventa pseudoaleatoare care se XOR-eaza mesajului clar
  - ctr este o ales in mod aleator la criptare
  - ctr se transmite in clar pentru ca este necesar la decriptare
  - Fk nu trebuie neaparat sa fie inversabili
  - Este paralelizabil
  - In plus, secventa pseudoaleatoare poate fi pre-procesata anterior decriptarii
  - CTR poate fi vazut si ca un sistem fluid nesincronizat

**Modurile CTR, OFB si CBC sunt CPA-sigure**  
**Modurile CBC, OFB si CTR folosesc un IV uniform aleator**  
**- asigura faptul ca Fk este mereu evaluat pe intrari diferite (previne situatia in care adversarul afla informatii la vederea de intrari identice)**

- Daca IV se repeta:**
- Pentru modurile OFB si CTR, intregul stream pseudoaleator (cu care se face xor pe mesaj) se repeta
  - Daca IV nu este uniform aleator (deci este predictibil), CTR este sigur dar CBC nu este sigur.

**Sisteme de criptare CPA-Sigure (ChosenPlainTextAttack)**

- Se pot cripta mai multe mesaje cu aceeasi cheie
- Cheia la fel de lunga ca mesajul

**Sisteme de criptare CCA-Sigure (ChosenCiphertextAttack)**

- Adversarul poate trimite orice mesaj m si primeste m criptat, poate trimite anumite mesaje c si primeste c decriptat
- Daca sistemul de criptare este nedeterminist, atunci oracolul de criptare foloseste de fiecare data o valoare aleatoare noua si neutilizata anterior.
- Consideram ca securitatea este impactata daca adversarul poate sa distinga intre criptarile a doua mesaje aleatoare
- CCA-Sigura  $\iff$  Un adversar nu poate determina care text clar a fost criptat cu o probabilitate semnificativ mai mare decat daca ar fi ghicit (in sens aleator, dat cu banul), chiar daca are acces la oracolele de criptare si decriptare.
- **Orice sistem CCA-Sigur este CPA-Sigur**
- Un sistem de criptare determinist **NU** poate fi CCA-Sigur
- orice schema de criptare care permite ca textele criptate sa fie modificate intr-un mod controlat nu poate fi CCA-sigura
- Securitate CCA  $\Rightarrow$  securitate CPA  $\Rightarrow$  securitate semantica
- Securitate - interceptare simpla  $\Rightarrow$  securitate - interceptare multipla

- AES – Advanced Encryption Standard**
- este folosit in multe standarde comerciale: IPsec, TLS, IEEE 802.11i (WPA2), SSH, Skype, etc.
  - AES este o retea de substitutie - permutare pe 128 biti care poate folosi chei de 128, 192 sau 256 biti
  - Lungimea cheii determina numarul de runde:

Lungime cheie (biti)	128	192	256
Număr runde	10	12	14
  - Foloseste o matrice de octeti  $4 \times 4$  numita stare
  - Starea initiala este mesajul clar ( $4 \times 4 \times 8 = 128$ )
  - Starea este modificata pe parcursul rundelor prin 4 tipuri de operatii: AddRoundKey, SubBytes, ShiftRows, MixColumns
  - Iesirea din ultima runda este textul criptat
  - Singurele atacuri netriviiale sunt asupra AES cu numar 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 exista un atac mai eficient decat cautarea exhaustiva (brute-force) pentru AES cu numar complet de runde.

**CBC Cu padding PKCS7**

- in urma decriptarii se obtin  $m1 || m2$
- se citeste octetul final b
- daca ultimii b octeti au toti valoarea b, atunci se scoate padding-ul si se obtine mesajul original m
- altfel intoarce mesajul padding gresit si cere retransmiterea mesajului
- Server-ul actioneaza ca un oracol de padding - adversarul ii trimite texte criptate si afla daca padding-ul este corect sau nu

- DES – Data Encryption Standard**
- DES este o retea de tip Feistel cu 16 runde si o cheie pe 56 biti
  - Procesul de derivare a cheii (key schedule) obtine o sub-cheie de runda  $k_i$  pentru fiecare runda pornind de la cheia master  $k$
  - Functiile de runda  $f_i(R) = f(k_i, R)$  sunt derivate din aceeasi functie principala  $f_b$  si nu sunt inversabile
  - Fiecare sub-cheie  $k_i$  reprezinta permutarea a 48 biti din cheia master
  - Intreaga procedura de obtinere a sub-cheilor de runda este fixa si cunoscuta, singurul secret este cheia master
    - ▶ Funcția  $\hat{f}$  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**

- Formeaza o parte esentiala din constructia DES
- DES devine mult mai vulnerabil la atacuri daca SBOX-urile sunt modificate usor sau daca sunt alese aleator
- Primul si ultimul bit din cei 6 de la intrare sunt folositi pentru a alege linia din tabel, iar bitii 2-5 sunt folositi pentru coloana; output-ul va consta din cei 4 biti aflati la intersectia liniei si coloanei alese.
- Modificarea unui bit de la intrare intotdeauna afecteaza cel putin doi biti de la iesire.
- DES are un puternic efect de avalansa generat de ultima, proprietate mentionata mai sus si de permutarile folosit

**Securitatea sistemului DES**

- Inca de la propunerea sa, DES a fost criticat din doua motive: 1. Spatiul cheilor este prea mic facand algoritmul vulnerabil la forta bruta; 2. Criteriile de selectie a SBOX-urilor au fost tinute secrete si ar fi putut exista atacuri analitice care explorau proprietatile matematice ale SBOX-urilor, cunoscute numai celor care l-au proiectat.
- Cu toate acestea.... Dupa 30 ani de studiu intens, cel mai bun atac practic ramane doar o cautare exhaustiva pe spatiul cheilor
- O cautare printre 256 chei este fezabila azi (dar netriviala);
- In 1977, un calculator care sa efectueze atacul intr-o zi ar fi costat 20.000.000\$
- In 2006, o echipa de cercetatori de la universitatile Bochum si Kiel din Germania a construit masina COPACABANA (Cost Optimized Parallel Code-Breaker) care permite gasirea cheii DES cu un timp de cautare mediu de mai putin de 7 zile, la un cost de 10.000\$.
- O alta problema a sistemului DES, mai putin importanta, este lungimea blocului relativ scurta (64 biti)
- Securitatea multor constructii bazate pe cifruri bloc depinde de lungimea blocului
- In modul de utilizare CTR, daca un atacator are 227 perechi text clar/text criptat, securitatea este compromisa cu probabilitate mare

**Criptanaliza diferentiala**

- Daca o diferentiala exista cu probabilitate  $p \gg 2^{-n}$ , cifrul bloc nu mai este permutare pseudoaleatoare
- Ideea este de a folosi multe diferentiale cu p usor mai mare decat  $2^{-n}$  pentru a gasi cheia secreta folosind un atac cu text clar ales
- Criptanaliza diferentiala a fost folosita cu succes pentru a ataca cifruri bloc (altele decat DES si AES), de pilda FEAL-8
- Pentru o functie aleatoare se asteapta ca  $p = 0.5$
- Singura vulnerabilitate practica DES este cheia scurt

**Meet in the middle**

- $K_1, K_2$  chei independente
- Lungimea totala a cheii este de 112 biti dar poate fi sparta in  $2^{56}$  deoarece pentru un  $(x,y)$  text clar, criptat cunoscute, pleaca si cripteaza  $x$  cu toate  $2^{56}$  posibile chei  $k_1$  apoi decripteaza  $y$  cu toate  $2^{56}$  posibile chei  $k_2$  iar la final daca in cele 2 multimi se gasesc elemente identice atunci valorile satisfac  $\Rightarrow y = F(k_1, k_2)(x)$  cu acele  $k_1$  si  $k_2$
- Complexitate timp  $O(2^n)$

**Criptare tripla**

- Trei chei independente sau doua chei independente si functia de criptare  $= F_{k_1}(F_{k_2}^{-1}(F_{k_1}(x)))$
- Prima varianta are lungimea cheii  $3n$  dar cel mai bun atac necesita timp  $2^{2n}$  (functioneaza atacul meet-in-the-middle);
- A doua varianta are lungimea cheii  $2n$  si cel mai bun atac necesita timp  $2^{2n}$ .

**Triplu DES (3DES)**

- Se bazeaza pe tripla invocare a lui DES folosind doua sau trei chei;
- Este considerat sigur si in 1999 l-a inlocuit pe DES ca standard;
- 3DES este foarte eficient in implementarile hardware (la fel ca si DES) dar totusi lent in implementari software;
- 3DES cu 3 chei inca se mai foloseste dar recomandarea este de a inlatura datorita lungimii mici a blocurilor si a faptului ca este lent
- Este popular in aplicatiile financiare si in protejarea informatiilor biometrice din pasapoartele electronice
- Singurele dezavantaje ar fi lungimea mica a blocurilor si faptul ca este destul de lent fiindca aplica DES de trei ori;
- Acestea au dus la inlocuirea lui ca standard cu AES
- DES-X este o varianta DES care rezista mai bine (decat DES) la forta brut
- DES-X foloseste doua chei suplimentare  $k_1, k_2$
- DES a fost sistemul simetric dominant de la mijlocul anilor '70 pana la mijlocul anilor '90
- DES cu cheia pe 56 biti poate fi spart relativ usor astazi prin forta bruta
- Insa, este foarte greu de spart folosind criptanaliza diferentiala sau liniara
- Pentru 3DES nu se cunoaste nici un atac practic

**MAC**

- Criptarea NU ofera integritate mesajelor
- Nu folositi criptarea cu scopul de a obtine autentificarea mesajelor
- Vom folosi un mecanism diferit, numit cod de autentificare a mesajelor – MAC (Message authentication code)
- Scopul lor este de a impiedica un adversar sa modifice un mesaj trimis fara ca partile care comunica sa nu detecteze modificarea
- Vom lucra in continuare in contextul criptografiei cu cheie secreta unde partile trebuie sa prestabileasca de comun acord o cheie secreta

MAC

- Un cod de autentificare a mesajelor (MAC) definit peste  $(\mathcal{K}, \mathcal{M}, \mathcal{T})$  este format dintr-un triplet de algoritmi polinomiali  $(\text{Gen}, \text{Mac}, \text{Vrfy})$  unde:
- 1.  $\text{Gen}(1^n)$ : este algoritmul de generare a cheii  $k$  (aleasă uniform pe  $n$  biți)
  - 2.  $\text{Mac} : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{T}$  este algoritmul de generare a tag-urilor  $t \leftarrow \text{Mac}_k(m)$ ;
  - 3.  $\text{Vrfy} : \mathcal{K} \times \mathcal{M} \times \mathcal{T} \rightarrow \{0, 1\}$  este algoritmul de verificare ce întoarce un bit  $b = \text{Vrfy}_k(m, t)$  cu semnificația că:
    - ▶  $b = 1$  înseamnă valid
    - ▶  $b = 0$  înseamnă invalid
- $a.f : \forall m \in \mathcal{M}, k \in \mathcal{K} \text{ Vrfy}_k(m, \text{Mac}_k(m)) = 1$ .
- Cazuri folosire MAC: cand cele doua parti care comunica partajeaza o cheie secreta in avans SAU cand avem o singura parte care autentifica o comunicare, interactionand cu ea insasi dupa un timp

Definiție

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

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

- MAC NU ofera nicio protectie la atacurile prin replicare ((in care un adversar copiaza un mesaj impreuna cu tag-ul aferent trimise de partile comunicante) Mai degraba, protectia impotriva replicarii trebuie facuta la nivel inalt de catre aplicatiile care folosesc MAC-uri

Constructia MAC-urilor sigure

- Functia MAC ar putea fi un PRF
- Fie  $F : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}^n$  o PRF. Definim un MAC în felul următor:
- ▶  $\text{Mac} :$  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);
  - ▶  $\text{Vrfy} :$  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).
- Daca F este PRF, constructia de mai sus reprezinta un cod de autentificare a mesajelor sigur (nu poate fi falsificat prin atacuri cu mesaj ales).
- Constructia prezentata anterior functioneaza doar pe mesaje de lungime fixa (PRF-urile sunt instantiate cu sisteme de criptare bloc care, cel mai adesea, accepta input-uri de 128 biti);
- insa in practica avem nevoie de mesaje de lungime variabila;
- Aratam cum putem obtine un MAC de lungime variabila pornind de la un MAC de lungime fixa

CBC-MAC

- O solutie mult mai eficienta este sa folosim CBC-MAC
- CBC-MAC este o constructie similara cu modul CBC folosit pentru criptare
- Folosind CBC-MAC, pentru un tag aferent unui mesaj de lungime  $l \cdot n$ , se aplica sistemul bloc doar de  $l$  ori, iar tag-ul are numai  $n$  biti.

- Daca F este o functie pseudoaleatoare, constructia de mai sus reprezinta un cod de autentificare a mesajelor sigur (nu poate fi falsificat prin atacuri cu mesaj ales) pentru mesaje de lungime  $l \cdot n$  pentru  $l$  fixat
- Constructia prezentata este sigura numai pentru autentificarea mesajelor de lungime fixa;
- $l$  fixat inseamna ca cele doua parti care comunica trebuie sa partajeze  $l$  in avans
- Avantajul acestei constructii fata de cea anterioara este ca ea poate autentifica mesaje de lungime mult mai mare;

Criptare în mod CBC	CBC-MAC
<ul style="list-style-type: none"><li>▶ IV este aleator pentru a obține securitate;</li><li>▶ toate blocurile <math>c_i</math> constituie mesajul criptat.</li></ul>	<ul style="list-style-type: none"><li>▶ IV = <math>0^n</math> este fixat pentru a obține securitate;</li><li>▶ doar ieșirea ultimului bloc constituie tag-ul (întoarcerea tuturor blocurilor intermediare duce la pierderea securității)</li></ul>

Constructii MAC

- CBC – MAC - folosit pe larg in industria bancar
- HMAC (Hash MAC) - pentru protocoale pe Internet: SSL, IPsec, SSH...

HMAC

- ipad si opad sunt doua constante de lungimea unui bloc mi
- ipad consta din byte-ul 0x5C repetat de atatea ori cat e nevoie
- opad consta din byte-ul 0x36 repetat de atatea ori cat e nevoie
- IV este o constanta fixata.

Combinari criptare si autentificare mesaje

- **Criptare-si-autentificare:** criptarea si autentificarea se fac independent.
- Combinatia aceasta nu este neaparat sigura; un MAC sigur nu implica nici un fel de confidentialitate;
- **Autentificare-apoi-criptare:** intai se calculeaza tag-ul t apoi mesajul si tag-ul sunt criptate impreuna
- Combinatia aceasta nu este neaparat sigura
- Se poate construi o schema de criptare CPA-sigura care impreuna cu orice MAC sigur nu poate fi CCA-sigura
- **Criptare-apoi-autentificare:** Combinatia aceasta este intotdeauna sigura; se foloseste in IPsec
- Desi folosim aceeasi constructie pentru a obtine securitate CCA si transmitere sigura a mesajelor, scopurile urmarite sunt diferite in fiecare caz

Securitate CCA vs Criptare autentificata

- Orice schema de criptare autentificata este si CCA-sigura dar exista si scheme de criptare CCA-sigure care nu sunt scheme de criptare autentificat
- Totusi, cele mai multe aplicatii de scheme de criptare cu cheie secreta in prezenta unui adversar activ necesita integritate



Criptare autentificata in practica

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

- Scheme de criptare autentificata: OCB, GCM, CCM, EAX
- TLS foloseste GCM

Functii Hash

- Acestea primesc ca argument o secventa de lungime variabila pe care o comprima intr-o secventa de lungime mai mica, fixata;
- Daca 2 valori au acelasi hash => coliziune
- In criptografie, o functie hash ramane o functie care comprima secvente de lungime variabila in secvente de lungime fixa, ins
- Daca in contextul structurilor de date este preferabil sa se minimizeze coliziunile, in criptografie acest lucru este impus;
- Daca in contextul structurilor de date coliziunile apar arbitrar (valorile sunt alese independent de functia hash), in criptografie coliziunile trebuie evitate chiar daca sunt cautate in mod voit (de catre adversar).
- NU exista functii hash fara coliziuni
- In aceste conditii, o functie hash impune ca determinarea unei coliziuni sa devina dificil

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

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

- **3 nivele de securitate**
- - Rezisenta la coliziuni – cea mai puternica
- - Rezisenta la a doua preimagine - presupune ca fiind dat x este dificil de determinat  $x' \neq x$  a.i.  $H(x) = H(x')$
- - Rezisenta la prima preimagine - presupune ca fiind dat  $H(x)$  este imposibil de determinat x.
- Rezisent la coliziuni => Rezisenta la a doua preimagine
- Rezisent la a doua preimagine => Rezisenta la prima imagine

Atacul zilei de nastere necesita ~  $2^{(l(n)/2)}$  evaluari

Transformarea Merkle-Damgard

- Numim functie de compresie o functie hash rezistenta la coliziuni de intrare de lungime fixa;
- Cu cat domeniul si codomeniul functiei sunt mai apropiate ca dimensiune, numarul de coliziuni scade => coliziunile devin mai dificil de determinat (co
- Scopul este sa construim o functie hash (cu intrare de lungime variabila), pornind de la o functie de compresie (de lungime fixa). Pentru aceasta se aplica transformarea Merkle-Damgard

Teoremă

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

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

Exemple functii hash:

MD5 (Message Digest 5):

- Definit in 1991 de R.Rivest ca inlocuitor pt MD4
- produce o secventa hash de 128 biti
- nesigur din 1996
- utilizat in versiuni mai vechi de Moodle

SHA (Secure Hash Algorithm):

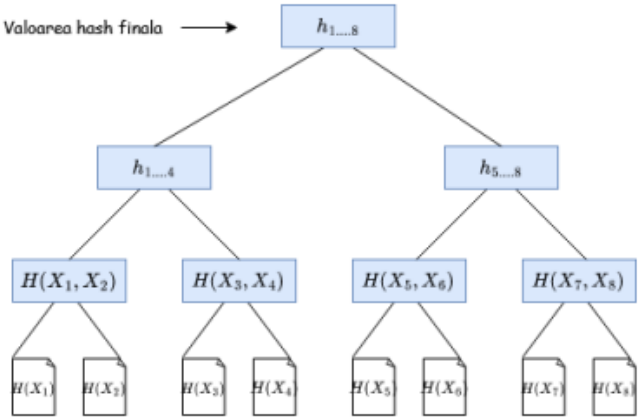
- o familie de functii hash publicate de NIST
- SHA-0 si SHA-1 sunt nesigure
- SHA-2 prezinta 2 variante: SHA-256 si SHA-512
- SHA-3 adoptat in 2012 pe baza Keccak

Gasire coliziuni:

- cel mai bun atac generic pentru gasirea de coliziuni ne spune ca avem nevoie de functii hash cu output-ul pe  $2n$  biti pentru securitate impotriva adversarilor care ruleaza in timp  $2^n$  spre deosebire de **sistemele de criptare bloc**, unde avem nevoie de  $n$  biti pentru securitate impotriva adversarilor care ruleaza in timp  $2^n$
- ex: Daca dorim ca gasirea de coliziuni sa necesite timp 2 128 atunci trebuie sa alegem functii hash cu output-ul pe 256 biti

Aplicatii functii hash:

- Amprentarea virusilor: scanner-ele de virusi 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 – Similar cu arbori de intervale pentru Hash( $x_1 \dots x_n$ )



Stocarea parolelor

- Greseala frecventa la register user: mesaj de tip: nume de utilizator existent, parola gresita
- Pentru stocarea parolelor se utilizeaza funct,iile hash
- Functiile hash sunt one-way (Daca stii H(pwd) nu poti determina pwd)
- Aceasta metoda de stocare a parolelor introduce deci avantajul ca nu ofera adversarului acces direct la parole, chiar daca acesta det,ine fi,sierul de parole

Atac de tip dictionar

- Adversarul poate insa sa verifice, pe rand, toate parolele cu probabilitate mare de aparit,ie. Acestea sunt de obicei cuvinte cu sens sau parole uzuale
- Pentru a minimiza ,sansele unor astfel de atacuri:
  - se blocheaza procesul de autentificare dupa un anumit numar de incercari nereu,site
  - se obliga utilizatorul sa foloseasca o parola care satisface anumite criterii : o lungime minima, utilizarea a cel put,in 3 tipuri de simboluri (litere mici, litere mari, cifre, caractere speciale)
- In caz de succes, adversarul determina parola unui singur utilizator
- Pentru a determina parolele mai multor utilizatori simultan, un adversar poate precalcu la valorile hash ale parolelor din dict,ionar
- Daca adversarul capata acces la fi,sierul de parole, atunci verifica valorile care se regasesc in lista precalcu la
- Toate conturile utilizatorilor pentru care se potrivesc valorile sunt compromise
- Atacul necesita capacitate de stocare mare: trebuie stocate toate perechile (pwd, H(pwd)) unde pwd este o parola din dict,ionar

Rainbow Tables

- introduc un compromis intre capacitatea de stocare ,si timp;
- f este o funct,ie de mapare a valorilor hash in parole
- Atent,ie! Funct,ia f nu este inversa funct,iei H (s-ar pierde proprietatea de unidirect,ionalitate a funct,iei hash)
- Se memoreaza doar capetele lant,urilor, valorile intermediare se genereaza la nevoie
- Daca t este suficient de mare, atunci capacitatea de stocare scade semnificativ
- Algoritmul de determinare a unei parole:
  - 1. Se cauta valoarea hash a parolei in lista de valori hash a tablei stocate. Daca se gase,ste, atunci lant,ul cautat este acesta ,si se trece la pasul 2. Daca nu se gase,ste, se reduce valoarea hash prin funct,ia de reducere f intr-o parola careia i se aplica funct,ia H ,si se reia cautarea de la pasul 1
  - 2. Se genereaza intreg lant,ul plecand de la valoarea init,iala stocata. Parola corespunzatoare este cea situata in lant, inainte de valoarea hash cautata.

Salting

- Pentru a evita atacurile precedente, se utilizeaza tehnica de salting;
- La inregistrare, se stocheaza pentru fiecare utilizator: (username,salt, H(pwd | |salt))
- salt este o secvent,a aleatoare de n bit,i, distincta pentru fiecare utilizator;
- Adversarul nu poate precalu la valorile hash inainte de a obt,ine acces la fi,sierul de parole decat daca folose,ste 2n valori posibile salt pentru fiecare parola;
- Atacurile devin deci impracticabile pentru n suficient de mare
- In plus, in practica se folosesc funct,ii hash lente
- Un alt avantaj introdus de tehnica de salting este ca de,si 2 utilizatori folosesc aceea,si parola, valorile stocate sunt diferite
- Prin simpla citire a fi,sierului de parole, adversarul nu i,si poate da seama ca 2 utilizatori folosesc aceea,si parola

Parole de unica folosinta

- Cazul extrem de schimbare periodica a parolei il reprezinta parolele de unica folosint,a;
- Utilizatorul folose,ste o lista de parole, la fiecare logare utilizand urmatoarea parola din lista;
- Aceasta lisa se calculeaza pornind de la o valoare x folosind o funct,ie hash H;

TOT DE MAI SUS ESTE CRIPTOGRAFIE SIMETRICA

Criptografia Asimetrica

- Criptografia cu cheie publica este introdusa de W.Diffie ,si M.Hellman in 1976 ca o solut,ie la problemele enumerate anterior
- Vom studia sisteme de criptare care folosesc chei diferite pentru criptare ,si decriptare - sisteme de criptare asimetrice;

Definiție

Un sistem de criptare asimetric definit peste (M,C), cu:

► M = spațiul textelor clare (mesaje)

► C = spațiul textelor criptate

este un triplet (Gen,Enc,Dec), unde:

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

a.î.  $\forall m \in M, (pk, sk) \text{ generate cu algoritmul } Gen(1^n)$

$Dec_{sk}(Enc_{pk}(m)) = m.$

- Foloseste o pereche de chei:
- Cheia publica pk este folosita pentru criptare
- Cheia secreta sk este folosita pentru decriptare
- Cheia publica este larg raspandita pentru a asigura posibilitatea de criptare oricui dore,ste sa transmita un mesaj catre entitatea careia ii corespunde
- Cheia secreta este privata ,si nu se cunoa,ste decat de entitatea careia ii corespunde

Continuare Criptografie Asimetrica

- Consideram (pentru simplitate) ca ambele chei au lungime cel put, in n bit, i

<b>Criptografia simetrică</b> <ul style="list-style-type: none"><li>▶ necesită secretizarea întregii chei</li><li>▶ folosește aceeași cheie pentru criptare și decriptare</li><li>▶ rolurile emițătorului și ale receptorului pot fi schimbate</li><li>▶ pentru ca un utilizator să primească mesaje criptate de la mai mulți emițători, trebuie să partajeze cu fiecare câte o cheie</li></ul>	<b>Criptografia asimetrică</b> <ul style="list-style-type: none"><li>▶ necesită secretizarea unei jumătăți din cheie</li><li>▶ folosește chei distincte pentru criptare și decriptare</li><li>▶ rolurile emițătorului și ale receptorului nu pot fi schimbate</li><li>▶ o pereche de chei asimetrice permite oricui să transmită informație criptată către entitatea căreia îi corespunde</li></ul>
<b>Avantaje</b> <ul style="list-style-type: none"><li>▶ număr mai mic de chei</li><li>▶ simplifică problema distribuirii cheilor</li><li>▶ fiecare participant trebuie să stocheze o singură cheie secretă de lungă durată</li><li>▶ permite comunicarea sigură pe canale publice</li><li>▶ rezolvă problema mediilor de comunicare deschise</li></ul>	<b>Dezavantaje</b> <ul style="list-style-type: none"><li>▶ criptarea asimetrică este mult mai lentă decât criptarea simetrică</li><li>▶ compromiterea cheii private conduce la compromiterea tuturor mesajelor criptate primite, indiferent de sursă</li><li>▶ necesită verificarea autenticității cheii publice (PKI rezolvă această problemă)</li></ul>

- Criptografia simetrica NU rezolva toate problemele criptografiei
- Criptografia asimetrica apare in completarea criptografiei simetrice

**Grup** -> Asociativitate, element neutru, element inversabil

: Ordinul lui  $\mathbb{Z}_{21}^*$  = 12 pentru că

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

- La criptografia asimetrica (cu cheie publica) construct,iile cunoscute se bazeaza pe probleme matematice dificile din teoria numerelorLa criptografia asimetrica (cu cheie publica) construct,iile cunoscute se bazeaza pe probleme matematice dificile din teoria numerelor
- In continuare vom introduce cateva doua considerate "dificile": problema factorizarii ,si problema logaritmului discret ,si vom prezenta funct,ii conjecturate ca fiind one-way bazate pe aceste probleme;

**Problema Factorizarii**

- Fiind dat un numar compus N, problema cere sa se gaseasca doua numere prime x1 ,si x2 pe n bit,i a,sa incat  $N = x_1 \cdot x_2$ ;
- Cele mai greu de factorizat sunt numerele care au factori primi foarte mari
- Pentru a putea folosi problema in criptografie, trebuie sa generam numere prime aleatoare in mod eficient
- Putem genera un numar prim aleator pe n bit,i prin alegerea repetata de numere aleatoare pe n bit,i pana cand gasim unul prim;

- Pentru eficient,a, ne intereseaza doua aspecte: 1. probabilitatea ca un numar aleator de n bit, i sa fie prim; 2. cum testam eficient ca un numar dat p este prim.
  - Pentru distribut,ia numerelor prime, se cunoa,ste urmatorul rezultat matematic: Pentru orice  $n > 1$ , proportia de numere prime pe n bit, i este de cel put, in  $1/3n$
  - Rezulta imediat ca daca testam  $t = 3n^2$  numere, probabilitatea ca un numar prim sa nu fie ales este cel mult  $e^{-n}$  , deci neglijabila.
  - Deci avem un algoritm in timp polinomial pentru gasirea numerelor prime
  - Cei mai eficient,i algoritmi sunt probabili,sti
  - Deocamdata nu se cunosc algoritmi polinomiali pentru problema factorizarii
  - Dar exista algoritmi mult mai buni decat fort,a bruta
  - Prezntam in continuare cat,iva algoritmi de factorizare
- RSA Challenge**
- In 1991, Laboratoarele RSA lanseaza 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 provocari, cate 1 pentru fiecare dimensiune (in bit,i) a lui N:
  - Exemple includ: RSA-576, RSA-640, RSA-768, . . . , RSA-1024, RSA-1536, RSA-2048;
  - Provocarea s-a incheiat oficial in 2007
  - Multe provocari au fost sparte in cursul anilor (chiar ,si ulterior inchiderii oficiale), insa exista numere inca nefactorizate:
  - RSA-1024, RSA-2048
- Problema logaritmului discret**
- Consideram G un grup ciclic de ordin q
  - Exista un generator  $g \in G$  a.i.  $G = \{g^0, g^1, \dots, g^{q-1}\}$
  - Echivalent, pentru fiecare  $h \in G$  exista un unic  $x \in \mathbb{Z}_q$  a.i.  $g^x = h$
  - x se nume,ste logaritmul discret al lui h in raport cu g ,si se noteaza  $x = \log_g h$
- Definiție**
- Spunem că problema logaritmului discret (DLP) este dificilă dacă pentru orice algoritm PPT  $A$  există o funcție neglijabilă  $negl$  așa încât
- $$Pr[DLog_A(n) = 1] \leq negl(n)$$
- Exista cateva clase de grupuri ciclice pentru care DLP este considerata dificila
  - Una dintre ele este clasa grupurilor ciclice de ordin prim (in aceste grupuri, problema este "cea mai dificila")
  - DLP nu poate fi rezolvata in timp polinomial in grupurile care nu sunt de ordin prim, ci doar este mai u,soar
  - In aceste grupuri cautarea unui generator ,si verificarea ca un numar dat este generator sunt triviale
  - DLP este considerata dificila in grupuri ciclice de forma  $\mathbb{Z}^*_p$  cu p prim
  - Insa pentru  $p > 3$  grupul  $\mathbb{Z}^*_p$  NU are ordin prim



Important Factorizare + Discret

- Cel mai rapid algoritm de factorizare necesita timp sub-exponential
- Problema logaritmului discret este dificila.

SHA-1

- output pe 160 biti
- 2005 - atac teoretic pentru gasirea coliziunilor; necesita 269 evaluari ale functiei hash
- 2017 - prima coliziune practica - 263 evaluari ale functiei hash
- atac pentru gasirea de coliziuni cu prefix - aprox. 267 evaluari

SHA-2

- prezinta doua versiuni: SHA-256 ,si SHA-512 in functie de lungimea output-ului
- nu se cunosc vulnerabilitati; atat SHA-2 cat ,si SHA-3 sunt siguri de folosit acolo unde rezistent,a la coliziuni este necesara

SHA-3

- Atacurile asupra MD5 si SHA-1 au impus necesitatea unei noi functii hash
- **Keccak** este castigatorul, SHA-2 ramane in continuare sigura
- Criterii selectie: Lungimea secventei de iesire: n = 224, 256, 384 ,si 512 biti, Alte dimensiuni ale secventei de iesire sunt optionale, Eficienta crescuta fata de SHA-2, Utilizare in HMAC, Rezistent,a la coliziuni, prima ,si a doua preimagine (conform cu atacurile generice, traditionale), Demonstratie de securitate, Parametrizabila, numar de runde variabil, Simplitate, claritate
- Keccak a fost gandit sa difere complet de constructiile existente (AES, SHA-2), foloseste **sponge functions**
- Principala componenta este permutarea f care accepta blocuri de 1600 biti.
- 2 faze : Absorbing phase: mesajul de intrare se sparge in blocuri de lungime r care se XOR-eaza la prima parte a starii, alternand cu aplicarea functiei f ; si squeezing phase: partea superioara a starii este returnata la iesire, alternand cu aplicarea functiei f ; numarul de iteratii depinde de numarul de biti necesari la iesire.

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)	≥ min(d, 128)	min(d, 128)
SHAKE256	d	min(d/2, 256)	≥ min(d, 256)	min(d, 256)

**Securitatea perfecta** NU este posibila la criptografie cu cheie publica

- Indistinctibilitatea in criptografia cu cheie publica este corespondenta notiunii similare din criptografia cu cheie secreta;

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

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

Teoremă

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

- Indistinctibilitate = securitate CPA

Criptare hibrida

- Criptarea cu cheie secreta este mult mai rapida decat criptarea cu cheie publica
- Pentru mesajele care sunt suficient de lungi, se foloseste criptare cu cheie secreta in tandem cu criptarea cu cheie publica
- Rezultatul acestei combinatii se numeste criptare hibrida ,si este folosita extensiv in practica

Pentru criptarea unui mesaj m, se urmeaza doi pasi:

- Expeditorul alege aleator o cheie k pe care o cripteaza folosind cheia publica a destinatarului, rezultand  $c1 = Enc_{pk}(k)$ ; Numai destinatarul va putea decripta k, ea ramanand secreta pentru un adversar;
- Expeditorul cripteaza m folosind o schema de criptare cu cheie secreta ( $Enc', Dec'$ ) cu cheia k, rezultand  $c2 = Enc'_k(m)$ ;
- Mesajul criptat este  $c = (c1, c2)$

Teoremă

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

- Pentru criptarea mesajelor lungi, in practica se foloseste criptarea hibrida
- Aceasta imbina avantajele criptarii simetrice ,si criptarii asimetrice

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

1. Ușor de calculat: Există un algoritm polinomial pentru calculul lui  $f$
2. Dificil de inversat: Pentru orice algoritm polinomial  $\mathcal{A}$ , există o funcție neglijabilă  $negl$  așa încât

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

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 numește radacina de ordin  $e$  a lui  $x$  modulo  $N$
- ▶ Dacă  $p$  și  $q$  se cunosc, atunci putem calcula  $\phi(N)$  și  $d = e^{-1} \bmod \phi(N)$
- ▶ Dacă  $p$  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  $N$
- ▶ Considerăm algoritmul 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  $\text{negl}$  așa încât

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

- In practica se foloseste  $e=3$  sau  $e=16$  la GenRSA pt exponentiere eficienta
- Pentru ca problema RSA sa poata fie dificila, trebuie ca  $N$ -ul ales in GenRSA sa fie dificil de factorizat in produs de doua numere prime;

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  $c = m^e \bmod N$ ;
  3. **Dec:** dată o cheie secretă  $(N, d)$  și un mesaj criptat  $c \in \mathbb{Z}_N$ , întoarce  $m = c^d \bmod N$ .
- ▶ Sistemul de criptare este corect pentru ca  $\text{Dec}_{sk}(\text{Enc}_{pk}(m)) = m$  astfel:  
 $(m^e)^d \bmod N = m^{ed \bmod \phi(N)} \bmod N = m^1 \bmod N = m$
- NU este CPA-Sigur, NU este CCA-Sigur
- NU este corect sa se utilizeze mai multe perechi de chei care folosesc acelasi modul
- **RSA este cel mai cunoscut ,si mai utilizat algoritm cu cheie public**
- Textbook RSA NU trebuie utilizat!

Padded RSA

- Ideea este sa se adauge un numar aleator (pad) la mesajul clar inainte de criptare
- Notam in continuare cu  $n$  parametrul de securitate (conform GenRSA)
  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  $c = (r||m)^e \bmod N$ ;
  3. **Dec:** dată o cheie secretă  $(N, d)$  și un mesaj criptat  $c \in \mathbb{Z}_N$ , calculează  $c^d \bmod N$  și întoarce ultimii  $l(n)$  biți.
    - ▶ 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 ( Public Key Cryptography Standard )

- martie 1998 - Laboratoarele RSA introduc PKCS #1 v1.5;
  - Folose,ste Padded RSA
  - Utilizat in HTTPS, SSL/TLS, XML Encryption
    - ▶ Notăm  $k$  lungimea modului  $N$  în bytes:  $2^{8(k-1)} \leq N < 2^{8k}$ ;
    - ▶ Mesajele  $m$  care se criptează se consideră multipli de 8 biți, de maxim  $k - 11$  bytes;
    - ▶ Criptarea se realizează astfel:  
$$(00000000||00000010||r||00000000||m)^e \bmod N$$
    - ▶  $r$  este ales aleator, pe  $k - D - 3$  bytes nenuli, unde  $D$  este lungimea lui  $m$  în bytes;
  - Se crede ca este CPA-sigur, nu e demonstrat
  - NU este CCA-Sigur
  - octombrie 1998 - Laboratoarele RSA introduc un nou standard PKCS demonstrat CCA-sigur in modelul ROM (Random Oracle Model) = **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  $m' = \text{OAEP}(m, r)$ , unde  $r$  este o secvență aleatoare (de lungime  $n$ );
  - ▶ RSA-OAEP criptează  $m \in \{0, 1\}^{n/2}$  folosind cheia publică  $(N, e)$  ca:  
$$(\text{OAEP}(m, r))^e \bmod N$$
  - ▶ RSA-OAEP decriptează  $c$  folosind cheia secretă  $(N, d)$  ca:  
$$(m, r) = \text{OAEP}^{-1}(c^d \bmod N)$$
  - Utilizarea RSA in practica: PKCS #1 v1.5 ,si PKCS #1 v2.0 (OAEP)

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 algoritmi care pot obține un timp mai bun la rulare decât forța brută, realizând un compromis spațiu-timp.
- ▶ 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 algoritmi 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  $g \in \text{mathbb{G}}$  generator, elementele lui  $\mathbb{G}$  sunt
$$1 = g^0, g^1, g^2, \dots, g^{q-1}, g^q = 1$$
- ▶ 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*)
$$g^0, g^t, g^{2t}, \dots, g^{\lceil q/t \rceil \cdot t}$$
- ▶ știm că  $h = g^x$  se află în unul din aceste intervale
- ▶ calculand, cu *baby-steps*, valorile
$$h \cdot g^1, h \cdot g^2, \dots, h \cdot g^t$$
- ▶ una din ele va fi egală cu unul din punctele marcate i.e.  $h \cdot g^i = g^{k \cdot t}$
- ▶ Complexitatea timp este  $\mathcal{O}(\sqrt{q} \cdot \text{polylog}(q))$  iar complexitatea spațiu este  $\mathcal{O}(\sqrt{q})$
- ▶ 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  $2^{160}$ .

- ▶ 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-exponențial în lungimea lui  $p$ .
- ▶ Această metodă seamănă cu algoritmul sitei pătratice pentru factorizare;
- ▶ Cel mai bun algoritm pentru DLP este sub-exponențial;
- ▶ Se pot construi funcții hash rezistente la coliziuni bazate pe dificultatea DLP;

Schimb de chei

- Am vazut ca bazele criptografiei cu cheie publica au fost puse de Diffie și Hellman în 1976 cand au introdus 3 primitive cu cheie publica diferite:
- ▶ **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.
- 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 comuna, secreta
- **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  $k_A = h_2^x$ .
  - ▶ 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
$$k_A = h_2^x = (g^y)^x = g^{xy}$$



Continuare Schimb de chei

- ▶ 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!

Computation Diffie-Hellman (CDH)

- ▶ 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).

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 **indistingibilă** 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  $A$  există o funcție neglijabilă  $negl$  așa încât:  
 $|Pr[A(\mathbb{G}, q, g, g^x, g^y, g^z) = 1] - Pr[A(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = 1]| \leq negl(n)$ ,  
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**.

- ▶ 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 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.

Schimbul de chei Diffie-Hellman nu rezista la atacuri active

Sistemul de criptare ElGamal

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  $m = c_2 \cdot c_1^{-x}$ .
- **NU** este determinist
  - ElGamal **NU** ramane sigur dacă problema DLP este simplă

Homomorfism

Un sistem de criptare care satisface  $Dec_{sk}(c_1 \cdot c_2) = Dec_{sk}(c_1) \cdot Dec_{sk}(c_2)$  se numește sistem de criptare **homomorfic**.  
(homomorfismul este deseori o proprietate utilă în criptografie)

- Se considera OK sa se foloseasca de mai multe ori aceeasi parametrii publici  $(G,q,g)$ . **Acest lucru nu este valabil si la RSA**
- 

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  
Însă poate fi modificat așa încât să fie CCA-sigur



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  $\mathbb{Z}_p$
- ▶ 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  $y_2 = -y_1$ ,  $P + Q = \mathcal{O}$
- ▶ altfel,  $P + Q = R$  de coordonate  $(x_3, y_3)$  care se calculează astfel:

$$\begin{aligned} x_3 &= [m^2 - x_1 - x_2 \bmod p] \\ y_3 &= [m(x_1 - x_3) - y_1 \bmod p] \end{aligned}$$

- ▶ 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  $P = Q$  și  $y_1 = 0$ , atunci  $P + Q = 2P = \mathcal{O}$
- ▶ Geometric, suma a două puncte  $P$  și  $Q$  se obține trasând o linie prin cele două puncte și găsim cel de-al treilea 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.
- ▶ În 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]:

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

- ▶ 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  $\text{card}(E(\mathbb{Z}_p)) = p$
- ▶ În practică, se folosesc curbe eliptice standardizate

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

- ▶ curba eliptică  $P-256$  (sau  $\text{secp256r1}$ ) 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  $y^2 = x^3 - 3x + B \bmod p$  iar  $p$ -ul ales astfel permite o implementare eficientă. Curbele  $P-384$  ( $\text{secp384r1}$ ) și  $P-521$  ( $\text{secp521r1}$ ) sunt definite în 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
- ▶  $\text{secp256k1}$  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 ecuația  $y^2 = x^3 + 7 \bmod p$ . Aceasta curbă eliptică este folosită în Bitcoin.

ECDLP – Problema logaritmului discret pe curbe eliptice

- ▶ 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  $Q = kP$ ;
- ▶ Notăție:  $\underbrace{P + P + \dots + P}_{s \text{ ori}} = sP$ .
- ▶ 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 algoritmi 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  $2^{160}$ .
- ▶ 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.

Criptografie pe curbe eliptice – ECC

- Dupa cercetari intensive, ECC pare foarte sigura, la fel de sigura precum RSA sau schemele bazate pe DLP
- Curbele eliptice sunt folosite pe larg ,si in standardele comerciale precum IPsec sau TLS
- ECC este adesea preferata in fat,a criptografiei cu cheie publica pentru sistemele incorporate precum dispozitivele mobile din motive de performanta
- implementarile pentru ECC sunt considerabil mai mici ,si mai rapide decat cele pentru RSA
- ECC cu chei pe 160-250 bit,i ofera cam acela ,si nivel de securitate precum RSA sau sistemele bazate pe DLP cu chei pe 1024-3072 bit,i

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

ECDLP este Dificila

ECCDH ( Elliptic curve computational Diffie-Hellman )

- O condit,ie minimala pentru ca protocolul sa fie sigur este ca ECDLP sa fie dificila in G;
- 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  $H_1, H_2 \xleftarrow{R} E(\mathbb{Z}_q)$ , 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).
- 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  $x, y, z \xleftarrow{R} \mathbb{Z}_q$

Sistemul transpus pe curbe eliptice pastreaza proprietatile sistemului initial

- Deci curba eliptica trebuie aleasa a.i. ECDLP si ECDDH sa fie dificile si sistemul ramane nedeterminist si homomorfic

Scheme de semnatura digitala

- Echivalentul MAC-urilor in criptografia cu cheie publica, dar exista cateva diferente importante intre ele
- O schema de semnatura digitala ii permite unui semnatar S care a stabilit o cheie publica pk sa semneze un mesaj in asa fel incat oricine care cunoaste cheia pk poate verifica originea mesajului (ca fiind S) si integritatea lui;

Avantaje semnaturi digitale peste MAC-uri

- Schemele de semnatura digitala sunt public verificabile ceea ce inseamna ca semnaturile digitale sunt transferabile - o terta parte poate verifica legitimitatea unei semnaturi si poate face o copie pentru a convinge pe altcineva ca aceea este o semnatura valida pentru m
- Schemele de semnatura digitala au proprietatea de non-repudiere - un semnatar nu poate nega faptul ca a semnat un mesaj;
- MAC-urile au avantajul ca sunt cam de 2-3 ori mai eficiente (mai rapide) decat schemele de semnatura digital

Definiție

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

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

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

Definiție

O semnătură  $\pi = (\text{Gen}, \text{Sign}, \text{Vrfy})$  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

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

- Paradigma "hash-and-sign" este sigură: înainte de semnare, mesajul trece printr-o funcție hash; varianta aceasta se folosește pe larg în practică;
  - $\sigma \leftarrow \text{Sign}_{sk}(H(m)); \text{Vrfy}_{pk}(H(m), \sigma) \stackrel{?}{=} 1$
- 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



Scheme de identificare / DSA / ECDSA

- ▶ sunt protocoale interactive care permit unei părți (Prover) să își demonstreze identitatea în fața unei alte părți (Verifier)
- ▶ sunt foarte importante ca building block pentru semnături digitale (dar, în sine, au aplicabilitate limitată)
- ▶ în continuare, abordare informală
- ▶ 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)
- ▶ Pentru a semna, Prover-ul execută singur protocolul generând challenge-ul pe baza unei funcții hash - folosește transformarea Fiat-Shamir
- ▶ Dacă problema logaritmului discret este dificilă, atunci schema de identificare Schnorr este sigură împotriva atacurilor pasive
- ▶ Dacă problema logaritmului discret este dificilă și  $H$  este modelată ca o funcție aleatoare, atunci semnătura Schnorr (obținută din schema de identificare Schnorr prezentată anterior, prin aplicarea transformării Fiat-Shamir) este sigură
- ▶ 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.
- ▶ 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 să nu fie predictibil. În caz contrar, cheia secretă  $x$  se poate afla (în ecuația  $s = k^{-1} \cdot (H(m) + xr) \bmod q$ , singura necunoscută este  $x$ ).
- ▶ De asemenea, folosirea aceluiași  $k$  pentru generarea a două semnături diferite duce la găsirea cheii secrete.
- ▶ O problemă a criptografiei cu cheie publică o reprezintă distribuția 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 anumită cheie publică;

Certificate si 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:

$$cert_{C \rightarrow B} = \text{Sign}_{sk_C}("Cheia\ lui\ Bob\ este\ pk_B")$$

- ▶ 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  $cert_{C \rightarrow B}$  a cărui validitate în raport cu  $pk_C$  Alice o verifică;
- ▶ 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 acestea sunt specificate într-o *infrastructură cu chei publice* (PKI-public key infrastructure) care permite distribuția la scară largă a cheilor publice;
- ▶ Există mai multe modele diferite de PKI, după cum vom vedea în continuare;
- ▶ 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;
- ▶ Modelul cu o singură CA nu este practic;
- ▶ În 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-urile 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;
- ▶ 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

$$cert_{B \rightarrow A} = \text{Sign}_{sk_B}("Cheia\ lui\ Alice\ este\ pk_A")$$

- ▶ Atunci când comunică cu Dan, Alice îi trimite

$$pk_A, cert_{B \rightarrow A}, pk_B, cert_{C \rightarrow B}$$

- ▶ De fapt,  $cert_{C \rightarrow B}$  conține, în afară de  $pk_B$  și afirmația "Bob este de încredere pentru a emite certificate"; astfel, Charlie îl delegă 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.

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
- Diferența 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
- În continuare vom prezenta o problemă care a primit multă atenție și care este considerată dificilă chiar și pentru calculatoarele cuantice. Aratăm apoi cum se poate construi un sistem de criptare cu cheie publică bazat pe dificultatea acelei probleme.

**Sistem de criptare bazat pe problema LWE (learning with errors) nu este sigur.** Insa el poate fi transformat într-un protocol sigursi adaptat ca un sistem de criptare adaugând “noise”, sub ipoteza problemei decizionale LWE

**Sistemul de criptare este CPA-sigur (chiar si pentru adversari cuantici) daca problema decizionala LWE este dificila.**

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  $(0^n, 0^s)$ .

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
- 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 și 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
- În continuare vom prezenta protocolul handshake din

- *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
  - 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
  - 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 privește server-ul, cheia K este calculată pe baza secretului y în fiecare sesiune, secret care este unul nou de fiecare data și care nu trebuie menținut între sesiuni; deci compromiterea server-ului (aflarea cheii lui secrete) nu duce la aflarea cheii K
  - în TLS 1.2, cheia secretă K ii este trimisă server-ului criptată cu cheia lui publică; 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 1.2

- Numar redus de runde
- 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