Criptografie Avansata

Mihai Prunescu*

Contents

*University of Bucharest, Faculty of Mathematics and Informatics; and Simion Stoilow Institute of Mathematics of the Romanian Academy, Research unit 5, P. O. Box 1-764, RO-014700 Bucharest, Romania.
mihai.prunescu@imar.ro, mihai.prunescu@gmail.com

Part I
Algebra

1 Grupuri

Definitie: Structura $(G, \cdot, 1)$ se numeste grup daca si numai daca satisface urmatoarele axiome:

1. Asociativitate: $\forall\, x, y, z\ (xy)z = x(yz)$.
2. Element neutru: $\forall\, x\ x1 = 1x = x$.
3. Inversabilitate: $\forall\, x\ \exists\, y\ xy = yx = 1$.

□

Definitie: Daca $(G, \cdot, 1)$ este un grup si $1 \in H \subseteq G$ o submultime, ea se numeste subgrup al lui G daca $(H, \cdot, 1)$ este grup. Notatia este $H \leq G$. □

Se constata usor ca daca $x, y \in G$ si $H \leq G$ subgrup, atunci $xH = yH$ sau $xH \cap yH = \emptyset$. De aici rezulta, de exemplu, ca in cazul in care G este finit, $|H|\,|\,|G|$.

Exemplu fundamental: Fie A o multime si $S(A)$ multimea functiilor bijective $f : A \to A$. Se observa ca $(S(A), \circ, 1 = id_A)$ este un grup, numit si grupul permutarilor lui A. Daca $|A| = n$, grupul se noteaza cu S_n. Se observa ca $|S_n| = 1 \cdot 2 \cdot \cdots \cdot n = n!$

Teorema lui Cayley: *Fie G un grup. Atunci $G \leq S(G)$.*

Demonstratie: Un element $g \in G$ se asociaza cu $f_g : G \to G$ data de $f_g(x) = gx$. Aceasta functie f_g este bijectiva, deci este in $S(G)$. Se arata usor ca $f_1 = id$, $f_g \circ f_h = f_{gh}$ si ca daca $f_{g_1} = f_{g_2}$ atunci $g_1 = g_2$. □

Definitie: Un subgrup $H \leq G$ se numeste subgrup normal daca si numai daca $\forall\, x \in G\ xHx^{-1} = H$. Subgrupul normal se noteaza $H \trianglelefteq G$.

Observam ca scufundarea data de Teorema lui Cayley nu il face pe G subgrup normal in $S(G)$ decat in cazuri foarte speciale.

Definitie: Fie doua grupuri G_1 si G_2. O functie $h : G_1 \to G_2$ se numeste homomorfism de grupuri daca $h(1) = 1$, si pentru orice $x, y \in G_1$, $h(xy) = h(x)h(y)$. Multimea $\{x \in G_1 \,|\, h(x) = 1\}$ se numeste nucleul lui h si se noteaza cu $Ker\ h$.

Teorema de Izomorfism: *Daca $h : G_1 \to G_2$ este un homomorfism de grupuri, atunci $Ker\ h \trianglelefteq G_1$ si:*

$$G_1/Ker\ h \simeq Im\ h \leq G_2.$$

Demonstratia acestei teoreme nu necesita alte cunostiinte decat definitiile prezentate mai sus. □

Definitie: Daca $A \subset G$, grupul generat de A, notat cu $\langle A \rangle$, este cel mai mic subgrup al lui G care-l contine pe A:

$$\langle A \rangle = \bigcap_{A \subseteq H \leq G} H.$$

Daca $a \in G$, $\langle a \rangle$ este subgrupul generat de a. Pentru grupul $(\mathbb{Z}, +, 0)$, $\mathbb{Z} = \langle 1 \rangle$. Se spune ca a are ordin finit daca si numai daca $\langle a \rangle$ este finit. Un grup se numeste ciclic daca este generat de un singur element. Grupurile ciclice finite sunt:

$$\langle a \rangle \simeq \mathbb{Z}/n\mathbb{Z},$$

unde $n = \operatorname{ord} a$ este cel mai mic $k > 0$ astfel incat $a^k = 1$.

Definitie: Grupul G este abelian sau comutativ daca pentru orice elemente $x, y \in G$, $xy = yx$.

Pentru grupurile abeliene se adopta notatia $(G, +, 0)$. Intr-un grup abelian, orice subgrup este normal. Observam ca daca $H \leq \mathbb{Z}/n\mathbb{Z}$ atunci exista $m \in \mathbb{N}$ cu $m|n$ si $H \simeq \mathbb{Z}/m\mathbb{Z}$.

Alte rezultate interesante legate de grupul de permutari:

Definitie: Functia $\varepsilon : S_n \to (\{-1, 1\}, \cdot)$ data de:

$$\varepsilon(\sigma) = \prod_{1 \le i < j \le n} \frac{\sigma(i) - \sigma(j)}{i - j}$$

se numeste signatura permutarilor si este un morfism de grupuri. $Ker\ \varepsilon := A_n$ este grupul altern, sau grupul permutarilor pare. Acest grup este subgrup normal in S_n dar la randul lui nu are alte subgrupuri normale, adica este un grup simplu. $|S_n/A_n| = 2$.

Definitie: Fie $a_1, \ldots, a_k$ obiecte distincte. Un ciclu $(a_1, a_2, \ldots, a_k)$ este o permutare c astfel incat $c(a_i) = a_{i+1}$ si $c(a_k) = a_1$, dar pentru orice obiect x diferit de $a_1, \ldots, a_k$, $c(x) = x$. Un ciclu de lungime 2 se numeste transpozitie.

Teorema: *Orice permutare $\sigma \in S_n$ se scrie ca produs de $\le n - 1$ transpozitii.*

Demonstratie: Fie $\sigma \in S_n$. Daca $\sigma(1) \neq 1$, consideram permutarea $(1, \sigma(1))\sigma$. Daca nu, il consideram din nou pe σ. In ambele cazuri, avem de a face acum cu o permutare τ astfel incat $\tau(1) = 1$. Daca $\tau(2) \neq 2$, consideram permutarea $(2, \tau(2))\tau$, daca nu, consideram τ. In tot cazul, avem de a face cu o permutare ρ care ii fixeaza pe 1 si pe 2. Dupa cel mult $n-1$ astfel de operatii, obtinem o permutare ι care ii fixeaza pe 1, 2, ..., $n-1$. Asta inseamna deci ca il fixeaza si pe n, deci $\iota = id$. Deci $t_k t_{k-1} \ldots t_1 \sigma = id$, und t_i sunt transpozitii. In final $\sigma = t_1 t_2 \ldots t_k$. □

In descompunerea in transpozitii descrisa mai sus, transpozitiile obtinute nu sunt in general disjuncte si nu comuta in general.

Teorema: *Orice permutare se descompune ca produs de cicluri disjuncte. In aceasta descompunere ciclurile comuta intre ele.*

Demonstratie: Pentru elementul 1 calculam $\sigma(1)$, $\sigma^2(1)$ si asa mai departe, pana cand $\sigma^k(1) = 1$. Avem deja un ciclu $(1, \sigma(1), \ldots, \sigma^{k-1}(1))$. Fie m cel mai mic intreg din $1, \ldots, n$ care nu este in orbita lui 1. Construim orbita lui m. Continual asa, pana cand toate elementele sunt intr-un ciclu. Prin constructie aceste cicluri sunt disjuncte. Fiind disjuncti, ei comuta intre ei, deoarece actioneaza pe multimi disjuncte. □

Exercitiu: Sa se demonstreze urmatoarele trei identitati cu cicluri si permutari:

$$(k, k+1) = (1, 2, \ldots, n)^{k-1}(1, 2)(1, 2, \ldots, n)^{1-k},$$

$$(i, j) = (j-1, j)(j-2, j-1)\ldots(i+1, i+2)(i, i+1)(i+1, i+2)\ldots(j-2, j-1)(j-1, j),$$

$$(a_1, a_2, \ldots, a_k) = (a_1, a_2)(a_2, a_3)\ldots(a_{k-1}, a_k),$$

unde $i < j$.

Exercitiu: Folosind identitatile din exercitiul precedent, aratati:

1. Transpozitiile $\{(k, k+1) \mid k = 1, \ldots, n-1\}$ genereaza impreuna intregul grup S_n.
2. Cele doua permutari $t = (1, 2)$ si $c = (1, 2, \ldots, n)$ genereaza impreuna intregul grup S_n.

2 Inele

Definitie: O structura $(R, +, \cdot, 0, 1)$ se numeste inel daca si numai daca respecta urmatoarele axiome:

1. $(R, +, 0)$ este grup abelian.
2. Asociativitatea inmultirii: $\forall\, x, y, z\ x(yz) = (xy)z$.
3. 1 este element neutru: $\forall\, x\ 1x = x1 = x$.

4. Distributivitatea inmultirii fata de adunare: $\forall\, x, y, z\ x(y+z) = xy+xz \wedge (y+z)x = yx+zx$.

Exemplu: Inelul numerelor intregi $(\mathbb{Z}, +, \cdot, 0, 1)$.

Definitie: O submultime $I \subseteq R$ se numeste ideal daca si numai daca $\forall\, a, b \in I\ a + b \in I$ si $\forall\, x \in R\ \forall\, a \in I\ ax \in I$.

Definitie: O functie $h : R_1 \to R_2$ este un homomorfism de inele daca si numai daca $h(1) = 1$ si $\forall\, a, b \in R\ h(a + b) = h(a) + h(b) \wedge h(ab) = h(a)h(b)$.

Teorema de izomorfism: Daca $h : R_1 \to R_2$ este un homomorfism de inele, atunci $Ker\ h$ este ideal in R_1 si

$$R_1/Ker\ h \simeq Im\ h \leq R_2 \text{ subinel.}$$

Exemplu esential: $I \subseteq \mathbb{Z}$ ideal daca si numai daca $I = n\mathbb{Z}$ pentru un $n \in \mathbb{N}$. Intr-adevar, daca $I \neq \{0\}$ luam $n_I = \min\{x \in I \,|\, x > 0\}$ si se arata ca $n_i\mathbb{Z} = I$. se observa de asemeni ca:

$$\begin{aligned} n\mathbb{Z} + m\mathbb{Z} &= \gcd(m, n)\mathbb{Z} \\ n\mathbb{Z} \cap m\mathbb{Z} &= \operatorname{lcm}(m, n)\mathbb{Z} \end{aligned}$$

Exemplu esential: $\mathbb{Z}/n\mathbb{Z}$ este un inel, numit si inelul de resturi modulo n. Intr-adevar, $(a + b) \bmod n = (a \bmod n + b \bmod n) \bmod n$ si asa mai departe.

Definitie: Fie R un inel. $R^\times = \{x \in R \,|\, \exists\, y \in R\ xy = 1\}$ este grupul multiplicativ al unitatilor lui R.

Observam ca $x \in (\mathbb{Z}/n\mathbb{Z})^\times$ daca si numai daca $\gcd(x, n) = 1$, adica sunt relativ prime. Exact aceeasi conditie este echivalenta cu faptul ca $\langle x \rangle_+ = \mathbb{Z}/n\mathbb{Z}$. Reamintim ca $\mathbb{Z}/n\mathbb{Z}$ se mai noteaza cu $\mathbb{Z}_n$.

Lema Chineza a Resturilor: Fie $n = p_1^{\alpha_1} \dots p_n^{\alpha_n}$ un numar natural nenul si descompunerea lui in factori primi. Atunci are loc urmatorul izomorfism de inele:

$$\mathbb{Z}/n\mathbb{Z} \simeq \mathbb{Z}/p_1^{\alpha_1}\mathbb{Z} \times \cdots \times \mathbb{Z}/p_k^{\alpha_k}\mathbb{Z}.$$

Demonstratie: Intr-adevar, daca $\gcd(m, n) = 1$ atunci $\mathbb{Z}_{mn} \simeq \mathbb{Z}_m \times \mathbb{Z}_n$ ca inele. Se vede ca:

$$p : \mathbb{Z} \to \mathbb{Z}_n \times \mathbb{Z}_m \ \ ; \ \ p(a) = (a \bmod n, b \bmod m)$$

are $Ker\ p = mn\mathbb{Z}$ si este surjectiv. □

O problema mai dificila este urmatoarea: daca avem resturile modulo n, m, ..., k; toate relativ prime, cum gasim un reprezentant pentru a modulo $mn \dots k$? Aceasta este o varianta efectiva a Teoremei Chineze a Resturilor este prezentata dupa ce facem cunostiinta cu notiunea de invers modular.

Deocamdata ne reamintim faptul ca $a \bmod n$ este inversabil in $\mathbb{Z}_n$ daca si numai daca $\gcd(a, n) = 1$.

Definitie: Indicatorul lui Euler este o functie $\varphi : \mathbb{N} \to \mathbb{N}$ astfel incat $\varphi(0) = 0$, $\varphi(1) = 1$ si

$$\varphi(n) = \#\{k \in \mathbb{Z}_n \mid \gcd(k, n) = 1\} = |\mathbb{Z}_n^\times|$$

Observam ca daca $\gcd(m, n) = 1$ atunci $\mathbb{Z}_{mn}^\times \simeq \mathbb{Z}_m^\times \times \mathbb{Z}_n^\times$, deci $\varphi(mn) = \varphi(m)\varphi(n)$.

Teorema: *Daca* $n = p_1^{\alpha_1} \dots p_n^{\alpha_n}$ *atunci:*

$$\varphi(n) = n(1 - \frac{1}{p_1}) \dots (1 - \frac{1}{p_n}).$$

Demonstratie:

$$\varphi(n) = \varphi(p_1^{\alpha_1}) \dots \varphi(p_n^{\alpha_n}) = (p_1^{\alpha_1} - p_1^{\alpha_1 - 1}) \dots (p_n^{\alpha_n} - p_n^{\alpha_n - 1}) =$$

$$= p_1^{\alpha_1} \dots p_n^{\alpha_n}(1 - \frac{1}{p_1}) \dots (1 - \frac{1}{p_n}) = n(1 - \frac{1}{p_1}) \dots (1 - \frac{1}{p_n}).$$

□

Teorema lui Euler: *Fie* $a, n > 0$ *cu* $\gcd(a, n) = 1$. *Atunci:*

$$a^{\varphi(n)} = 1 \bmod n.$$

Demonstratie: $|\mathbb{Z}_n^\times| = \varphi(n)$, $a \in \mathbb{Z}_n^\times$, $\text{ord}(a) | \varphi(n)$. □

Teorema lui Fermat: *Daca* p *prim,* $p \nmid a$*, atunci* $a^{p-1} = 1 \bmod p$.

Aceasta rezulta direct din Teorema lui Euler.

Observatie: Inelul numerelor intregi $\mathbb{Z}$ este euclidian, adica exista o functie norma $f : \mathbb{Z} \to \mathbb{N}$ astfel incat:

$$\forall\, a, b\ \exists\, r, q \quad q \geq 0 \wedge r \geq 0 \wedge 0 \leq f(r) < f(b) \wedge a = bq + r.$$

Intr-adevar, $\mathbb{Z}$ satisface aceasta conditie cu norma $f(b) = |b|$. Se spune ca in $\mathbb{Z}$ functioneaza impartirea cu rest.

Cu ajutorul impartirii cu rest se poate folosi Algoritmul lui Euclid pentru calcularea celui mai mare divizor comun $\gcd(a, b)$. In cazul in care $\gcd(a, b) = 1$ algoritmul se foloseste in forma extinsa pentru a calcula inversul modular $b^{-1} \bmod a$, adica inversul lui b in grupul unitatilor $\mathbb{Z}_a^\times$. Vom ilustra algoritmul printr-un exemplu.

Fie $a = 100$ si $b = 17$. Algoritmul lui Euclid functioneaza in modul urmator:

$$\begin{aligned} 100 &= 5 \cdot \underline{17} + \underline{15} \\ \underline{17} &= 1 \cdot \underline{15} + \underline{2} \\ \underline{15} &= 7 \cdot \underline{2} + \underline{1} \\ \underline{2} &= 2 \cdot \underline{1} + 0 \end{aligned}$$

Ultimul rest nenul este $\gcd(a, b)$. In cazul nostru $\gcd(100, 17) = 1$. Acum veti intelege de ce caturile si resturile au fost scrise subliniat. In partea a doua a algoritmului aceste caturi si resturi joaca rolul unor variabile literale intr-un proces de backwards substitution, dupa cum urmeaza. Calculul de mai jos este mod100.

$$1 = \underline{15} - 7\underline{2} = \underline{15} - 7(\underline{17} - \underline{15}) = 8 \cdot \underline{15} - 7 \cdot \underline{17} = 8 \cdot (-5) \cdot \underline{17} - 7 \cdot \underline{17} = (-47) \cdot 17 = 53 \cdot 17.$$

Deci $17^{-1} \bmod 100 = 53$.

Exista multe implementari ale Algoritmului lui Euclid. Urmatoarea implementare este foarte rapida, in special cu scrierea binara a numerelor naturale:

$g = 1$;
while ($a \bmod 2 = 0$ **and** $b \bmod 2 = 0$) **do** $a = a/2$; $b = b/2$; $g = 2g$, **end**;
while $a \neq 0$ **do**
while $a \bmod 2 = 0$ **do** $a = a/2$;
while $b \bmod 2 = 0$ **do** $b = b/2$;
(acum ambii sunt impari!)
if $a \geq b$ **then** $a = (a - b)/2$ **else** $b = (b - a)/2$;
end
return $g \cdot b$;

In sfarsit a venit timpul sa enuntam:

Teorema Chineza a Resturilor, varianta efectiva: *Fie* $m_1, \ldots, m_r \geq 2$ *cu* $\gcd(m_i, m_j) = 1$ *pentru* $i \neq j$. *Se dau conditiile* $x \bmod m_i = a_i$. *Fie:*

$$\begin{aligned} M &= m_1 m_2 \ldots m_r, \\ M_i &= M/m_i, \\ y_i &= M_i^{-1} \bmod m_i, \\ x &= \sum_{i=1}^{r} a_i M_i y_i \bmod M. \end{aligned}$$

In acest caz x *satisface conditiile date, deci este o solutie a sistemului de congruente.*

Demonstratie: Intr-adevar $x \bmod m_i = a_i M_i y_i \bmod m_i = a_i M_i M_i^{-1} \bmod m_i = a_i$. □

Exemplu: Cautam x astfel incat:

$$\begin{aligned} x &= 5 \bmod 7, \\ x &= 3 \bmod 11, \\ x &= 10 \bmod 13. \end{aligned}$$

Calculam $M = 7 \cdot 11 \cdot 13 = 1001$, $M_1 = 11 \cdot 13 = 143$, $M_2 = 7 \cdot 13 = 91$, $M_3 = 7 \cdot 11 = 77$. Mai departe $y_1 = 143^{-1} \bmod 7 = 3^{-1} \bmod 7 = 5$, $y_2 = 91^{-1} \bmod 11 = 3^{-1} \bmod 11 = 4$ si $y_3 = 77^{-1} \bmod 13 = 12^{-1} \bmod 13 = (-1)^{-1} \bmod 13 = -1 \bmod 13 = 12$. In sfarsit:

$$x = (5 \cdot 143 \cdot 5 + 3 \cdot 91 \cdot 4 + 10 \cdot 77 \cdot 12) \bmod 1001 = 894.$$

□

Urmatorul algoritm se va dovedi de o importanta capitala:

Algoritmul de exponentiere rapida. Pentru a calcula rapid a^b se foloseste faptul ca:

$$a^b = a^{\left(\sum_{b_i \in \{0,1\}} b_i 2^i\right)} = \prod_{b_i = 1} a^{2^i}.$$

Numerele de forma a^{2^i} se obtin prin ridicare succesiva la patrat:

$$a \rightsquigarrow a^2 \rightsquigarrow a^4 \rightsquigarrow a^8 \rightsquigarrow \cdots \rightsquigarrow a^{2^i} \rightsquigarrow a^{2^{i+1}} \rightsquigarrow \ldots$$

Acest algoritm este cu atat mai bun, cand se calculeaza $a^b \bmod c$ deoarece atunci toate numerele sunt marginite de c iar numarul de operatii este $\mathcal{O}(\log b)$.

Iata si o teorema foarte frumoasa, deocamdata fara demonstratie:

Teorema: *Grupul multiplicativ* $(\mathbb{Z}_n^*, \cdot, 1)$ *este ciclic daca si numai daca* $n \in \{2, 4, p^k, 2p^k\}$, *unde* p *reprezinta numerele prime impare.*

Exemplu: $\mathbb{Z}_9^\times = \{2, 4, 8, 7, 5, 1\}$, valori care sunt exact puterile succesive ale generatorului $g = 2$.

3 Corpuri

Definition: O structura $(K, +, \cdot, 0, 1)$ este un corp daca este un inel in care orice element diferit de 0 este inversabil la inmultire. Cu alte cuvinte, $(K, +, 0)$ este grup abelian, $(K \setminus \{0\}, \cdot, 1)$ este grup iar inmultirea $\cdot$ este distributiva in raport cu adunarea $+$.

Exemple: $(\mathbb{C}, +, \cdot, 0, 1)$ este corpul numerelor complexe. Multimile $\mathbb{R}$ si $\mathbb{Q}$ sunt subcorpuri. Corpul cuaternionilor $\mathbb{H}$ nu este comutativ. Pentru orice numar prim p, inelul $\mathbb{Z}_p = \{0, 1, \ldots, p-1\}$ este un corp. Acestea nu sunt toate corpurile finite, ci doar corpurile prime.

Definitie: Caracteristica unui corp este numarul minim $n \in \mathbb{N}$ astfel incat in corp $n \cdot 1 = 0$. Caracteristica infinita se numeste prin traditie caracteristica 0. Corpurile $\mathbb{H}$, $\mathbb{C}$, $\mathbb{R}$ si $\mathbb{Q}$ au caracteristica 0.

Nu este greu de observat ca o caracteristica finita este intotdeauna un numar prim. Fie K un corp de caracteristica $c = ab$ unde $a \geq 2$ si $b \geq 2$. Se observa ca:

$$(1 + \cdots + 1) + \cdots + (1 + \cdots + 1) = 0,$$

unde fiecare paranteza este o suma de a de 1 si sunt exact b paranteze. Daca dam factor comun, obtinem:

$$(1 + \cdots + 1 \text{ } a \text{ termeni})(1 + \cdots + 1 \text{ } b \text{ termeni}) = 0.$$

Dar corpurile fiind domenii de integritate, acest lucru este posibil doar daca una din paranteze este 0. Asta contravine cu minimalitatea lui $c = ab$.

Corpul $\mathbb{Z}_p$ are caracteristica p.

Fie K un corp de caracteristica p. K va contine elementele 1, $1 + 1$, $1 + 1 + 1$, ... Deci K il contine pe $\mathbb{Z}_p$. Cum ambele sunt corpuri, K este spatiu vectorial peste $\mathbb{Z}_p$. Presupune acum ca corpul K este finit. Fiind spatiu vectorial peste $\mathbb{Z}_p$, grupul aditiv $(K, +, 0)$ este izomorf cu un produs finit $\mathbb{Z}_p \times \cdots \times \mathbb{Z}_p$ si are p^k elemente.

Iata mai jos, fara demonstratie, cateva rezultate despre corpurile finite:

1. *Orice corp finit este comutativ. (Wedderburn)*
2. *Doua corpuri finite cu acelasi numar de elemente sunt izomorfe. Acest fapt justifica notatia $\mathbb{F}_{p^k}$ pentru corpul cu p^k elemente. $\mathbb{F}_p = \mathbb{Z}_p$.*
3. *$\mathbb{F}_{p^s} \subseteq \mathbb{F}_{p^r}$ daca si numai daca $s \mid r$.*

Exemplu: Corpul $\mathbb{F}_4$. Polinomul $f \in \mathbb{F}_2[X]$ dat de $f = X^2 + X + 1$ nu are solutii in $\mathbb{F}_2$ deoarece $f(0) = f(1) = 1$. Fie ω un simbol cu proprietatea ca $\omega^2 + \omega + 1 = 0$. In caracteristica 2, asta inseamna ca $\omega^2 = \omega + 1$. In inelul de polinoame $\mathbb{F}_2[X]$, polinomul ireductibil f genereaza un ideal prim, deci maximal, deoarece inelul este euclidian. Asadar definim corpul $\mathbb{F}_4$ in modul urmator:

$$\mathbb{F}_4 = \mathbb{F}_2[X]/(f) = \mathbb{F}_2[\omega] = \{0, 1, \omega, \omega + 1\}.$$

Ce grup multiplicativ este generat de ω? Calculam:

$$\omega,\ \omega^2 = \omega + 1,\ \omega^3 = \omega(\omega + 1) = \omega + 1 + \omega = 1.$$

Deci $\langle\omega\rangle = \mathbb{F}_4 \setminus \{0\}$, asadar grupul multiplicativ al lui $\mathbb{F}_4$ se dovedeste a fi ciclic. Teorema urmatoare arata ca acest fapt este general. El are o importanta cruciala in criptografia actuala.

Teorema: *Fie K un corp comutativ si fie G un subgrup finit al grupului $K^\times = (K \setminus \{0\})$. Atunci grupul G este ciclic. In particular, daca corpul K este finit, atunci grupul lui multiplicativ $K^\times$ este ciclic.*

Demonstratie: Fie h numarul de elemente al lui G. Vrem sa aratam ca exista un element de ordin h in G. Fie $h = p_1^{r_1} \dots p_h^{r_k}$ descompunerea lui h in factori primi. Observam ca pentru orice $i = 1, \dots, k$ exista un $x_i \in G$ astfel incat:

$$x_i^{\frac{h}{p_i}} \neq 1,$$

pentru ca altfel polinomul $X^{h/p_i} - 1$ ar avea un numar de solutii mai mare decat gradul, ceea ce nu este posibil intr-un corp. Definim elementul:

$$y_i = x_i^{\frac{h}{p_i} r_i}.$$

Aratam ca $\text{ord}(y_i) = p_i^{r_i}$. Intr-adevar $y_i^{p_i^{r_i}} = x_i^h = 1$, deci $\text{ord}(y_i) \mid p_i^{r_i}$. Presupunem ca $\text{ord}(y_i) = p_i^s$ cu $s < r_i$. Dar stim ca:

$$y_i^{p_i^{r_i - 1}} = x_i^{\frac{h}{p_i}} \neq 1,$$

ori $p_i^s \mid p_i^{r_i - 1}$, contradictie.

Fie $y = y_1 y_2 \dots y_k$. Cum ordinele elementelor y_i sunt prime intre ele, ordinul produsului este produsul ordinelor, deci ordinul lui y este h si y il genereaza pe G. □

Exemplu: $\mathbb{Z}_7^* = \{1, 2, 3, 4, 5, 6\} = \langle 3 \rangle$. Intr-adevar, sirul puterilor lui 2 mod 7 este 2, 4, 1, dupa care se repeta periodic. Pentru 3, perioada este 6. Numarul de generatori ai grupului ciclic este $\varphi(6) = \varphi(3)\varphi(2) = (3-1)(2-1) = 2$. Dar generatorii nu vor fi elementele care il genereaza aditiv de $\mathbb{Z}_6$. De exemplu 3 nu il genereaza aditiv pe $\mathbb{Z}_6$ dar il genereaza multiplicativ pe $\mathbb{Z}_7^*$. □

Observatie: Desi corpurile cu p^k elemente sunt toate izomorfe, ele pot fi definite folosind diferite polinoame ireductibile de grad k peste $\mathbb{F}_p$. De aceea, sunt mai multe reprezentari pentru aceleasi elemente respectiv operatii. In criptografie se folosesc explicit exact aceste reprezentari. De aceea, mentionarea lui p^k nu este suficienta fara mentionarea polinomului ireductibil de grad k care este folosit.

4 Functii in corpuri finite

Cel mai simplu corp finit este $\mathbb{F}_2 = \mathbb{Z}_2 = \{0\}$, cu urmatoarele operatii:

$+$	0	1
0	0	1
1	1	0

$\cdot$	0	1
0	0	0
1	0	1

Din punctul de vedere al logicii matematice, adunarea $+$ este functia logica XOR iar inmultirea $\cdot$ este conjunctia logica $\wedge$. Mai putem introduce negatia si disjunctia:

$\neg$	0	1
	1	0

$\vee$	0	1
0	0	1
1	1	1

Teorema: *Orice functie* $f : \{0,1\}^k \to \{0,1\}$ *se exprima folosind* $\vee$, $\wedge$ *si* $\neg$ *intr-o expresie.*

Demonstratie:

$$f = \bigvee_{\substack{(\varepsilon_1, \dots, \varepsilon_n) \\ f(\varepsilon_1, \dots, \varepsilon_n) = 1}} x_1^{\varepsilon_1} \wedge \dots \wedge x_n^{\varepsilon_n},$$

unde prin conventie $x^0 = \neg x$ si $x^1 = x$. $\square$

Exemplu: $x + y = x^0y^1 \vee x^1y^0 = (\neg x \wedge y) \vee (x \wedge \neg y)$.

Observam ca $x \wedge y = \neg(\neg x \vee \neg y)$, astfel incat putem reprezenta orice functie folosind numai negatia si disjunctia. Pe de alta parte $x \vee y = \neg(\neg x \wedge \neg y)$ deci alternativ putem folosi numai negatia si conjunctia pentru a exprima orice functie. Se poate merge chiar mai departe. Sa consideram operatia $\mid$ numita si NAND sau Sheffer's Stroke:

$$x \mid y = x^0y^0 \vee x^0y^1 \vee x^1y^0.$$

Se observa ca:

$$\neg x = (x \mid x),$$
$$x \vee y = (x \mid x) \mid (y \mid y).$$

Asadar orice functie se poate exprima folosind numai functia lui Sheffer.

In sfarsit, cum $x \wedge y = xy$ si $\neg x = 1 + x$, obtinem urmatorul rezultat: *Orice functie* $f : \{0,1\}^k \to \{0,1\}$ *poate fi exprimata printr-un polinom.* Acest fapt este general valabil pentru corpurile finite.

Teorema: *Fie* $\mathbb{F}$ *un corp finit. Orice functie* $f : \mathbb{F}^k \to \mathbb{F}$ *se reprezinta printr-un polinom.*

Demonstratiile se bazeaza pe interpolare. Interpolarea este metoda cu care, dat un corp K si k perechi de valori (x_i, y_i) unde valorile x_i sunt distincte, gasete un polinom f de grad $k-1$ astfel incat $f(x_i) = y_i$. Metoda se poate generaliza la mai multe variabile.

Demonstratia 1: Fie $\varepsilon \in \mathbb{F}$ o valoare oarecare. Definim polinomul $g_\varepsilon : \mathbb{F} \to \mathbb{F}$ prin:

$$g_\varepsilon(x) = \frac{\prod\limits_{a \neq \varepsilon}(x - a)}{\prod\limits_{a \neq \varepsilon}(\varepsilon - a)}.$$

Acest polinom are proprietatea ca:

$$g_\varepsilon(x) = \begin{cases} 1, & x = \varepsilon, \\ 0, & x \neq \varepsilon. \end{cases}$$

Fie $(\varepsilon_1, \ldots, \varepsilon_k) \in \mathbb{F}^k$ un tuplu oarecare. Definim polinomul $g_{(\varepsilon_1,\ldots,\varepsilon_k)}$ in modul urmator:

$$g_{(\varepsilon_1,\ldots,\varepsilon_k)}(x_1, \ldots, x_k) = g_{\varepsilon_1}(x_1)g_{\varepsilon_2}(x_2)\ldots g_{\varepsilon_k}(x_k).$$

Acest polinom ia valoarea 1 doar pentru $(x_1, \ldots, x_k) = (\varepsilon_1, \ldots, \varepsilon_k)$ si 0 in rest. In final, putem reprezenta functia f printr-un polinom:

$$f(x_1, \ldots, x_k) = \sum_{(\epsilon_1,\ldots,\epsilon_k)} g_{(\varepsilon_1,\ldots,\varepsilon_k)}(x_1, \ldots, x_k) \cdot f(\varepsilon_1, \ldots, \varepsilon_k).$$

Demonstratia 2: Ideea este asemanatoare, doar ca functia $g_\varepsilon(x)$ se construieste in alt mod. Fie n numarul de elemente ale corpului finit $\mathbb{F}$. Grupul multiplicativ $\mathbb{F} \setminus \{0\}$ are $n-1$ elemente, deci:

$$x^{n-1} = \begin{cases} 1, & x \neq 0, \\ 0, & x = 0. \end{cases}$$

Functia $g_\varepsilon(x)$ se defineste:

$$g_\varepsilon(x) = 1 - (x - \varepsilon)^{n-1}.$$

Restul demonstratiei este la fel. $\square$

5 Criptare lineara

Fie $\mathcal{A}$ un alfabet cu $|\mathcal{A}| = n$ elemente. Putem identifica $\mathcal{A}$ cu elemente ale unui inel finit R. Singura conditie este ca inelul sa fie mai mare decat alfabetul sau egal cu el. Fie $k \geq 1$. Un text in $\mathcal{A}$ se imparte in blocuri de lungime k iar acestea se cripteaza linear:

$$c : \mathcal{A}^k \to \mathcal{A}^k \text{ data de } c(a_1, \ldots, a_k) = M \cdot \begin{pmatrix} a_1 \\ \vdots \\ a_k \end{pmatrix}$$

unde $M \in \mathcal{M}_{k\times k}(R)$ este inversabila, adica exista o matrice M^{-1} astfel incat $M^{-1}M = I_k$.

Ne reamintim ca matricea inversa are elemente a_{ij} de forma:

$$a_{ij} = (-1)^{i+j} \det(M)^{-1} \det(M_{ij}).$$

Teorema: M este inversabila daca si numai daca $\det(M) \in R^\times$, adica este inversabil in R.

Exemple: Fie $|\mathcal{A}| = 26$ si blocuri de lungime 2, adica $x_1x_2 \rightsquigarrow y_1y_2$. Identificam $\mathcal{A}$ cu $\mathbb{Z}_{26}$. Operatia:

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} 6 & 2 \\ 5 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \bmod 26$$

nu este buna pentru criptare liniara deoarece $\gcd(26, \det(M)) = 2$, deci M nu este inversabila. Sa se gaseasca blocuri diferite x_1x_2 si $x_1'x_2'$ care au acceasi criptare y_1y_2.

Pentru operatia:

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} 6 & 1 \\ 5 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \bmod 26$$

gasiti regula de decriptare.

6 Curbe eliptice

Definitie: O curba eliptica peste $\mathbb{Z}_p$ este multimea perechilor $(x, y) \in \mathbb{Z}_p \times \mathbb{Z}_p$ cu:

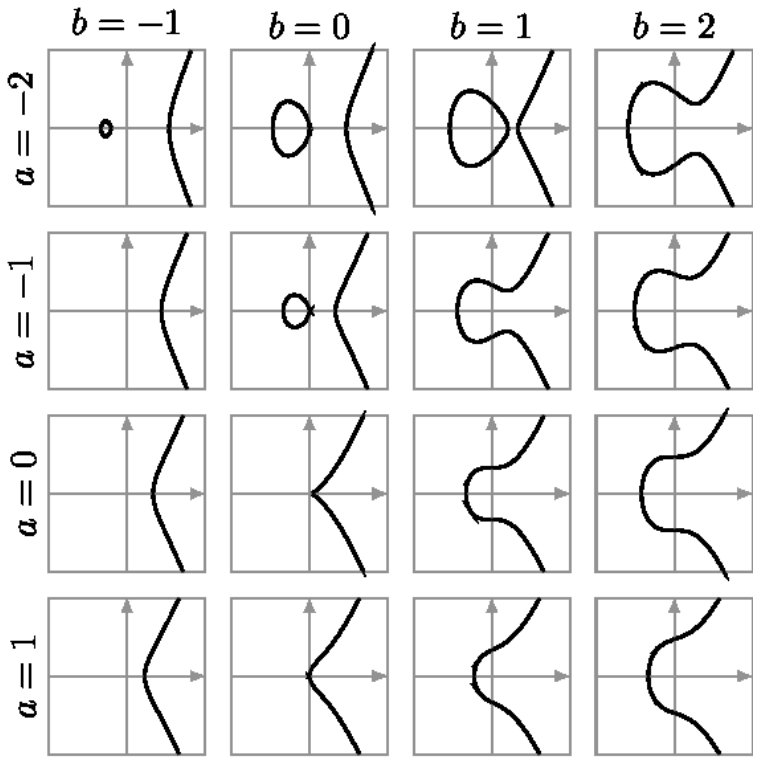
$$y^2 = x^3 + Ax + B \bmod p,$$

impreuna cu punctul de la infinit O, unde $A, B \in \mathbb{Z}_p$ si discriminantul:

$$4A^3 + 27B^2 \neq 0 \bmod p.$$

Aceasta multime se noteaza cu $E(\mathbb{Z}_p)$.

Curbe eliptice se pot defini pe orice corp, iar definitia generala este mai complicata. Curbele definite aici sunt cele in forma speciala Weierstraß. In prima figura vedeti cum depinde curba de parametrii in cazul real.



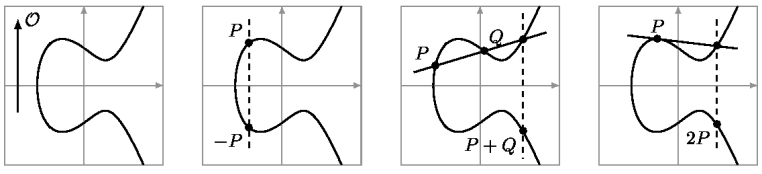
$$y^2 = x^3 + a \cdot x + b$$

O proprietate a curbelor eliptice este structura lor de grup. Operatia de grup se defineste in cazul real in modul urmator: dreapta care uneste punctele P si Q va intersecta curba de gradul 3 intr-un al treilea punct. Suma $P + Q$ este punctul simetric corespunzator acestui al treilea punct de intersectie, relativ la axa de simetrie al curbei. Daca efectuam $P + P$, se considera tangenta in P in locul secantei PQ. Punctul $-P$ este simetric cu punctul P relativ la axa de simetrie a curbei. Cum dreapta $P(-P)$ nu mai intersecteaza curba nicaieri, se considera ca $P + (-P) = O$. Asadar punctul de la infinit O este elementul neutru al grupului.

Cu ajutorul geometriei analitice putem deduce urmatoarele reguli de calcul pentru operatia de grup:

$$P + O = O + P = P,$$

$$P = (x, y), \quad Q = (x, -y) \quad \longrightarrow \quad P + Q = O,$$



Neutral element $\mathcal{O}$ | Inverse element $-P$ | Addition $P + Q$ "Chord rule" | Doubling $P + P$ "Tangent rule"

$$P + Q = -R, \quad R = (x_3, y_3),$$

$$x_3 = (m^2 - x_1 - x_2) \bmod p,$$

$$y_3 = m(x_2 - x_3) - y_1 \bmod p,$$

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

Teorema: *Pentru orice corp K, $(E(K), +, O)$ este un grup abelian. In cazul in care $K = \mathbb{Z}_p$ iar $p > 3$ este un numar prim, exista $n_1, n_2 > 1$ astfel incat:*

$$E(\mathbb{Z}_p) \simeq \mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2},$$

si $n_2 | n_1$, $n_2 | (p-1)$.

Asadar daca $n_2 = 1$, grupul $E(\mathbb{Z}_p)$ este ciclic. In practica se cauta puncte Pcare genereaza un subgrup ciclic de ordin prim. Urmatoarea teorema este de asemeni de interes:

Teorema lui Hasse:

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

Exemplu: Pentru corpul $\mathbb{Z}_7$ analizam curba eliptica E data de ecuatia $y^2 = x^3 + 2x + 1$. Calculam intai puterile modulo 7 ale elementelor:

x	0	1	2	3	4	5	6
x^2	0	1	4	2	2	4	1
x^3	0	1	1	6	1	6	6

Daca $x = 0$, $y^2 = 1$, deci $y = 1$ sau $y = 6$. Daca $x = 1$, $y^2 = 4$, deci $y = 2$ sau $y = 5$. Pentru $x = 2, 3, 4, 5, 6$, $y^2 = 6, 3, 5$ si nu obtinem puncte pe curba eliptica. Asadar:

$$E = \{O, (0,1), (0,6), (1,2), (1,5)\}.$$

Cum E are 5 elemente si exista un singur grup de 5 elemente, care este ciclic, rezulta ca oricare element diferit de O este generator al grupului. □

Exemplu: Curba E data de $y^2 = x^3 + 2x + 2 \bmod 17$. Punctul $P = (5,1)$ genereaza aditiv urmatoarea multime:

$$\langle P \rangle = \{(5,1), (6,3), (10,6), (3,1), (9,16), (16,13), (0,6), (13,7), (7,6),$$

$$(7,11), (13,10), (0,11), (16,4), (9,1), (3,16), (10,11), (6,14), (5,16), O\}$$

Grupul are 19 elemente si este ciclic. Fiecare element diferit de O este un generator. Se observa ca pentru fiecare punct P in randul 1 il gasim pe $-P$ in randul 2. □

Part II
Criptografie simetrica

7 Securitate perfecta

In acest paragraf se introduce limbajul probabilitatilor conditionate. $p(A|B)$ este probabilitatea evenimentului A conditionata de evenimentul B. Se stie ca:

$$p(A|B) = \frac{p(A \cap B)}{p(B)}.$$

Definitie: Se noteaza cu M multimea textelor clare, plain texts, cu K multimea cheilor de criptare si cu C multimea textelor criptate.

Definitie: Criptarea este o functie $Enc : M \times K \to C$. Decriptarea este o functie $Dec : C \times K \to M$. Conditia criptarii cu cheie simetrica este:

$$\forall\, m \in M\ \forall\, k \in K\ Dec_k(Enc_k(m)) = m.$$

Definitie: Se spune ca perechea (Enc, Dec) are securitate perfecta daca si numai daca:

$$\forall\, m \in M\ \forall\, c \in C\ p(m|c) = p(m).$$

Cu alte cuvinte, a il vedea pe c nu confera nicio informatie despre m.

In lema urmatoare, multimile K, C si M sunt considerate multimi finite:

Lema: Daca securitatea este perfecta,

$$|K| \geq |C| \geq |M|.$$

Demonstratie: Enc_k injectiva pentru orice $k \in K$ fixat, deci $|C| \geq |M|$.

Pentru orice $c \in C$, $p(c) > 0$, altfel il putem exclude pe c. Deci pentru orice $m \in M$ si pentru orice $c \in C$,

$$p(c|m) = p(c) > 0.$$

Deci pentru orice $m \in M$ si orice $c \in C$, exista $k \in K$ cu $Enc_k(m) = c$. Deci $|K| \geq |C|$. □

Teorema lui Shannon: Daca $|M| = |C| = |K|$ atunci (Enc, Dec) are securitate perfecta daca si numai daca sunt satisfacute ambele conditii de mai jos:

1. Orice cheie este folosita cu probabilitate de $1/|K|$.
2. $\forall\, m \in M\ \forall\, c \in C\ \exists!\, k \in K\ Enc_k(m) = c$.

Necesitatea celor doua conditii: Presupunem ca (Enc, Dec) are securitate perfecta. Pentru orice $m \in M$ si $c \in C$ exista $k \in K$ astfel incat $Enc_k(m) = c$. Dar $|C| = |K|$ deci $|\{Enc_k(m) \,|\, k \in K\}| = |K|$. Asadar pentru $m \in M$ si $c \in C$ exista o **unica** $k \in K$ astfel incat $Enc_k(m) = c$.

Acum vrem sa aratam ca orice cheie e folosita cu probabilitate egala. Fie $|K| = n$, $M = \{m_i \,|\, 1 \leq i \leq n\}$ si fixam $c \in C$. Securitate perfecta inseamna $p(m_i|c) = p(m_i)$, deci:

$$p(m_i) = p(m_i|c) = \frac{p(c|m_i)p(m_i)}{p(c)} = \frac{p(k_i)p(m_i)}{p(c)}.$$

Deci pentru orice i, $p(k_i) = p(c)$. Cheile se folosesc cu probabilitate egala.

Suficienta celor doua conditii: Presupunem ca cele doua conditii sunt indeplinite. Vrem sa aratam ca $p(m|c) = p(m)$. Fixam $m \in M$ si $c \in C$.

Deoarece cheile sunt folosite cu probabilitate egala,

$$p(c) = \sum_{k \in K, m \in M} p(k)p(m = Dec_k(c)) = \frac{1}{|K|} \sum_{k \in K, m \in M} p(m = Dec_k(c)).$$

Dar pentru orice m si c exista o unica cheie k astfel incat $Enc_k(m) = c$. Deci:

$$\sum_{k \in K, m \in M} p(m = Dec_k(c)) = \sum_{m \in M} p(m) = 1.$$

Fie din nou $n = |K|$. Deci $p(c) = 1/n$ iar pentru $c = Enc_k(m)$, $p(c|m) = p(k) = 1/n$. Conform cu legea lui Bayes a probabilitatii conditionate:

$$p(m|c) = \frac{p(m)p(c|m)}{p(c)} = \frac{p(m) \cdot 1/n}{1/n} = p(m).$$

□

Exemplu: Fie $\mathcal{A} = \mathcal{A}_{26} = \{A, B, C, \ldots, Z\}$. Atunci $|K| = |M| = |C| = 26^n$ si

$$c = m + k \bmod 26,$$

cu adunarea pe litere.

Exemplu: Vernam's Code, OTP. (One Time Pad) $\mathcal{A} = \{0, 1\}$, $|K| = |M| = |C| = 2^n$ si

$$c = m \oplus k,$$

unde $\oplus$ este adunarea din $\mathbb{F}_2$ si de face pe litere.

Observatie: A nu se folosi de mai multe ori aceeasi cheie. In acest caz:

$$c_1 \oplus c_2 = m_1 \oplus k \oplus m_2 \oplus k = m_1 \oplus m_2,$$

deci se afla o parte din informatia criptata foarte usor.

8 Entropie

Definitie: Fie X o variabila aleatoare, care ia valori $(x_i)_{1 \leq i \leq n}$ cu probabilitatile $p_i = p(x_i)$. Entropia lui X se defineste in modul urmator:

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i,$$

cu conventia $p = 0 \to p \log_2 p = 0$.

Exemplu: Daca la orice intrebare iti raspund numai cu *da*, atunci $p_1 = 1$, $p_2 = 0$ si

$$H(X) = -1 \log_2 1 - 0 \log_2 0 = 0,$$

adica dialogul cu mine nu iti ofera informatie. Daca raspunsurile mele au o probabilitate de $1/2$, atunci:

$$H(X) = (-\log_2 \frac{1}{2} - \log_2 \frac{1}{2})\frac{1}{2} = 1,$$

adica iti ofer 1 bit de informatie la fiecare raspuns.

Proprietati ale entropiei:

1. $H(X) \geq 0$.
2. $H(X) = 0 \leftrightarrow \exists! i \;\; p_i = 1 \wedge \forall j \neq i \;\; p_j = 0$.
3. $\forall i \; p_i = \frac{1}{n} \to H(X) = \log_2 n$.

4. Inegalitatea lui Jensen

$$\sum_{i=1}^{n} a_i = 1 \rightarrow \sum_{i=1}^{n} a_i \log_2 x_i \leq \log_2 \Big(\sum_{i=1}^{n} a_i x_i \Big),$$

cu egalitate daca $x_1 = \cdots = x_n$. Aceasta inegalitate exprima faptul ca functia log are un grafic convex.

Teorema: *Daca X ia n valori posibile,*

$$0 \leq H(X) \leq \log_2 n.$$

Demonstratie:

$$H(X) = -\sum p_i \log_2 p_i = \sum p_i \log_2 \frac{1}{p_i} \leq \log_2 \sum p_i \frac{1}{p_i} = \log_2 n.$$

□

Definitie: Fie X si Y doua variabile aleatoare. Definim entropia conditionata:

$$H(X|y) = -\sum_x p(X = x|Y = y) \log_2 p(X = x|Y = y),$$

$$H(X|Y) = \sum_y p(Y = y) H(X|y).$$

Definitie: Fie X si Y doua variabile aleatoare.
Definim entropia comuna. Daca $r_{ij} = p(X = x_i \wedge Y = y_j)$,

$$H(X, Y) = -\sum_{i=1}^{n} \sum_{j=1}^{m} r_{ij} \log_2 r_{ij}.$$

Proprietati:

1. $H(X, Y) \leq H(X) + H(Y)$ cu egalitate daca si numai daca sunt independente.
2. $H(X, Y) = H(Y) + H(X|Y)$.
3. $H(X|Y) \leq H(X)$ cu egalitate daca si numai daca sunt independente.

In criptografie urmatoarele relatii sunt importante:
$H(M|K, C) = 0$, adica daca stim mesajul criptat si cheia, aflam mesajul clar.
$H(C|M, K) = 0$, adica daca stim mesajul si cheia, stim mesajul criptat.

$$\begin{aligned} H(K, M, C) &= K(M, K) + H(C|M, K) =, \text{ dar } H(C|M, K) = 0, \\ &= H(M, K) = \\ &= H(K) + H(M) \text{ deoarece } K \text{ si } M \text{ sunt independente.} \end{aligned}$$

$$\begin{aligned} H(K, M, C) &= K(K, C) + H(M|K, C) =, \text{ dar } H(M|K, C) = 0, \\ &= H(K, C) \end{aligned}$$

In final:

$$H(K,C) = H(K) + H(M).$$

Definitie: $H(K|C)$ se numeste *key equivocation* si reprezinta cantitatea de nesiguranta legata de cheie dupa ce ni se arata un text criptat.

$$H(K|C) = H(K,C) - H(C) = H(K) + H(M) - H(C).$$

Exemplu: $M = \{a,b,c,d\}$, $C = \{1,2,3,4\}$, $K = \{1,2,3,4\}$ cu $p(a) = 0,25$, $p(b) = p(d) = 0,3$, $p(c) = 0,15$. De asemeni $p(k_1) = p(k_3) = 0,25$, $p(k_2) = 0,5$. Criptarea este:

	a	b	c	d
k_1	3	4	2	1
k_2	3	1	4	2
k_3	4	3	1	2

Calculam:

$$\begin{aligned}
p(1) &= p(k_1)p(d) + p(k_2)p(b) + p(k_3)p(c) = 0,2625\\
p(2) &= p(k_1)p(c) + p(k_2)p(d) + p(k_3)p(d) = 0,2625\\
p(3) &= p(k_1)p(a) + p(k_2)p(a) + p(k_3)p(b) = 0,2625\\
p(4) &= p(k_1)p(b) + p(k_2)p(c) + p(k_3)p(a) = 0,2125\\
H(M) &\sim 1,9527\\
H(K) &\sim 1,5\\
H(C) &\sim 1,9944\\
H(K|C) &\sim 1,9527 + 1,5 - 1,9944 = 1,4583
\end{aligned}$$

Deci, daca vedem un mesaj criptat, mai trebuie sa gasim cam $1,45$ biti informatie despre cheie. Asta este foarte putin, deci sistemul este foarte nesigur.

Daca fiecare mesaj clar ar avea probabilitatea de $1/4$ si fiecare cheie ar avea probabilitatea de $1/3$, atunci fiecare mesaj cifrat ar avea probabilitatea de $3(1/3)(1/4) = 1/4$. In acest caz entropiile sunt:

$$\begin{aligned}
H(M) &= -4 \cdot \frac{1}{4} \log_2 \frac{1}{4} = 2\\
H(K) &= -3 \cdot \frac{1}{3} \log_2 \frac{1}{3} = \log_2 3\\
H(C) &= -4 \cdot \frac{1}{4} \log_2 \frac{1}{4} = 2\\
H(K|C) &= 2 + \log_2 3 - 2 = \log_2 3 \sim 1,5849 > 1,4583
\end{aligned}$$

Deci cu o distributie uniforma de probabilitate, sistemul de criptare devine mai sigur! □

Fie L un limbaj natural, si fie H_L entropia pe litera, adica informatia pe litera. Pentru random strings cu alfabetul englez avem $H = \log_2 26 \sim 4,70$. Deci $H_L \leq 4,70$. Daca ne uitam la probabilitatile de aparitie ale literelor in limba engleza:

$$p(A) = 0,082; \ldots; p(E) = 0,127; \ldots; p(Z) = 0,001;$$

atunci $H_L \leq H(p) = 4,14$ biti de informatie pe litera. In realitate insa Q este intotdeauna urmat de U, TH este foarte frecvent, etc. Deci mai bine consideram grupuri de doua litere. Fie M^2 variabila aleatoare a digramelor.

$$H(M^2) = \sum_{i,j} p(M = i, M' = j) \log(M = i, M' = j) \sim 7,12$$

si $H_L \leq H(M^2)/2 = 3,56$.

Definitie: Entropia limbajului natural L este:

$$H_L = \lim_{n\to\infty} \frac{H(M^n)}{n}$$

si este estimata la limba engleza ca $1,0 \leq H_L \leq 1,5$.

Deci o litera in engleza necesita cam 5 biti pentru a fi codificata dar contine in context cel mult 1,5 biti de informatie. Asta inseamna ca limbajele naturale sunt foarte redundante. Iata un exemplu:

*On** up** a t**e t**re **s a **rl **ll** Sn** Wh**e.*

Definitie: Definim redundanta limbajului:

$$R_L = 1 - \frac{H_L}{\log_2 |M|}$$

Daca se ia o valoare medie de $H_L = 1,25$, redundanta in engleza va fi de:

$$R_L = 1 - \frac{1,25}{\log_2 26} \sim 0,75.$$

Deci putem comprima texte in aceasta limba de la 10 MB la 2,5 MB.

Acum urmeaza niste consideratii despre cheile false. Fie $c \in C$ un mesaj criptat de lungime $|c| = n$. Consideram multimea:

$$K(c) = \{k \in K \mid Ent_k(c) \text{ "are sens"}\}.$$

Atunci $|K(c)| - 1$ este numarul cheilor false.

Numarul mediu de chei false, sau piste false in decriptare, este:

$$s_n = \sum_{c\in C} p(c)(|K(c)| - 1) = \sum_{c\in C} p(c)|K(c)| - 1.$$

Pentru n mare si considerand $|M| = |K|$. Aplicand inegalitatea lui Jensen obtinem:

$$\log_2(s_n + 1) = \log_2 \sum_{c\in C} p(c)|K(c)| \geq \sum_{c\in C} p(c) \log_2 |K(c)|.$$

Deci:

$$\log_2(s_n + 1) \geq \sum_{c\in C} p(c)H(K|c) = H(K|C^n) = H(K) + H(M^n) - H(C^n) \sim$$
$$\sim H(K) + nH_L - H(C^n) = H(K) - H(C^n) + n(1 - R_L) \log_2 |M| \geq$$
$$\geq H(K) - n \log_2 |C| + n(1 - R_L) \log_2 |M| = H(K) - nR_L \log_2 |M|.$$

deoarece $H(C) \leq n \log_2 |C|$ si $|M| = |C|$. Asadar obtinem:

$$s_n \geq \frac{|K|}{|M|^{nR_L}} - 1.$$

Definitie: Distanta de unicitate n_0 este acea valoare a lui n la care numarul mediu asteptat de piste false devine 0.

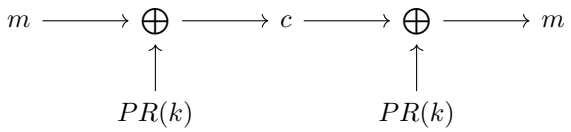
$$n_0 \sim \frac{\log_2 |K|}{R_L \log_2 |M|}.$$

Daca luam $|M| = 26$, $|K| = 26! \sim 4 \cdot 10^{26}$ si $R_L = 0,75$ obtinem $n_0 \sim 25$.

Deci pentru $|c| \geq 25$ se presupune exista o unica descifrare cu sens. Daca comprimam datele inainte de a le transmite, atunci $n_0 \sim |k|/0 \sim \infty$ deci atacatorul ar avea o munca mult mai dificila ca sa gaseasca o cheie cu sens.

9 Linear feedback shift registers

Securitatea perfecta nu este practica deoarece cheia are aceeasi lungime cu mesajul, trebuie aleasa la intamplare si schimbata de fiecare data. In loc de aceasta se folosesc generatori de siruri pseudo-aleatoare. Un asemenea pseudo-random generator primeste ca argument o cheie k mult mai scurta decat mesajul m. Generatorul produce un sir pseudo-random de lungimea dorita care se aduna binar cu m. Decriptarea se face dupa aceeasi schema:



Cum se obtine un generator pseudo-random? Orice program determinist, cu input k, va produce un sir care la un moment dat devine periodic. Asadar conditiile sunt:

- Perioada sa fie cat mai lunga cu putinta.

- Sirul sa fie "imprevizibil".

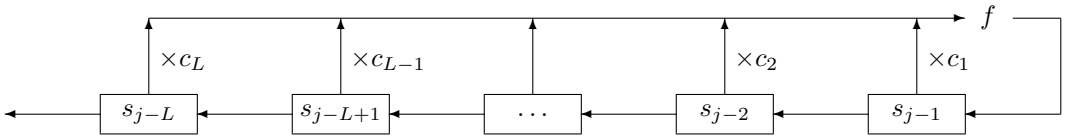
O prima idee este sa alegem $p \in \mathbb{N}$ foarte mare astfel incat $\langle 10 \rangle_{\times} = \mathbb{Z}_p^{\times}$. In acest caz, $1/p$ are o fractie periodica iar perioada are lungime p. Proprietatile pseudo-random trebuie testate de la caz la caz.

O alta posibilitate este sa alegem un numar prim p si un numar k astfel incat $\langle k \rangle_{\times} = \mathbb{Z}_p^{\times}$. Sirul pseudo-random este sirul puterilor $(k^n)_{n \in \mathbb{N}}$ cu o codificare potrivita - de exemplu toate numerele scrise binar, si cuvintele binare concatenate. Evident, si concatenarea cifrelor zecimale rezulta intr-un sir pseudo-random, ca si sirul abstract, cu valori in $\mathbb{Z}_p^{\times}$.

In continuare ne ocupam de registrele liniare cu feedback, linear feedback shift registers sau LFSR.

Definitie: Fie $L \geq 1$ lungimea registrului, $c_1, \ldots, c_L \in \{0,1\}$ coeficientii registrului, $[s_0, \ldots, s_{L-1}]$ starea initiala a registrului, unde $s_i \in \{0,1\}$. Secventa produsa este $(s_n)_{n \in \mathbb{N}}$ unde:

$$s_j = f(s_{j-1}, \ldots, s_{j-L}) = c_1 s_{j-1} \oplus c_2 s_{j-2} \oplus \cdots \oplus c_L s_{j-L}, \quad \forall\, j \geq L.$$



Un LFSR are o perioada $N < 2^L - 1$. Acest lucru va rezulta din urmatoarele considerente.

Unui LSFR i se asociaza urmatoarea matrice de conexiune $M \in \mathcal{M}_{L \times L}(\mathbb{F}_2)$ si urmatoarea tranzitie de stare:

$$\begin{pmatrix} s_{j-L+1} \\ s_{j-L+2} \\ \vdots \\ s_{j-1} \\ s_j \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 1 \\ c_L & c_{L-1} & c_{L-2} & \ldots & c_1 \end{pmatrix} \begin{pmatrix} s_{j-L} \\ s_{j-L+1} \\ \vdots \\ s_{j-2} \\ s_{j-1} \end{pmatrix}$$

Definitie: Polinomul de conexiune al unui LFSR este polinomul:

$$C(X) = 1 + c_1 X + \cdots + c_L X^L \in \mathbb{F}_2[X].$$

Definitie: Fie $\mathbb{F}$ un corp finit, $C(X) \in \mathbb{F}[X]$ un polinom ireductibil si $\mathbb{G} = \mathbb{F}[X]/(C)$ este corpul de descompunere al lui $C(X)$. Polinomul $C(X)$ se numeste polinom primitiv daca o solutie ω a lui $C(X)$ genereaza grupul ciclic multiplicativ $\mathbb{G}^{\times}$.

Observam ca aplicarea matricii de conexiune vectorilor $\vec{v} \in \mathbb{F}_2^L$ este o operatie izomorfa cu inmultirea elementelor din $\mathbb{F}_{2^N}$ cu un element special ω^{-1}, unde ω este solutie a polinomului $C(X)$. Se deosebesc urmatoarele cazuri:

In orice caz se exclude starea initiala $s_0 = 0$, deoarece aceasta genereaza sirul de valori constant.

Cazul singular: $c_L = 0$. Pentru multe stari initiale sirul de valori devine periodic abia dupa un segment initial. Exista cicluri periodice disjuncte de lungimi diferite.

Cazul reductibil: $c_L = 1$ si $C(X)$ este reductibil peste $\mathbb{F}_2$. In acest caz pentru toate starile initiale sirul este periodic dar exista cicluri periodice disjuncte de lungimi diferite.

Cazul ireductibil neprimitiv: $c_L = 1$ si $C(X)$ este ireductibil peste $\mathbb{F}_2$ dar nu este polinom primitiv. In acest caz sirul este periodic pentru orice stare initiala. Exista cicluri disjuncte dar toate au aceeasi lungime.

Cazul primitiv: $c_L = 1$ si $C(X)$ este polinom primitiv. In acest caz exista un singur ciclu de lungime maxima $|\mathbb{G}^\times| = 2^L - 1$.

Observatie: LSFR este nesigur pentru plain text attack. Daca se cunoate L si se cunosc $2L$ valori, se pot afla coeficientii din sistemul urmator peste corpul $\mathbb{F}_2$:

$$\begin{pmatrix} s_{L-1} & s_{L-2} & \cdots & s_0 \\ s_L & s_{L-1} & \cdots & s_1 \\ \vdots & \vdots & \ddots & \vdots \\ s_{2L-2} & s_{2L-3} & \cdots & s_{L-1} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_L \end{pmatrix} = \begin{pmatrix} s_L \\ s_{L+1} \\ \vdots \\ s_{2L-1} \end{pmatrix}$$

Pentru a zadarnici asemenea atacuri si pentru a mari lungimea perioadei, se combina mai multe LSFR folosind functii nelineare, ca de exemplu:

$$F(x_1, x_2, x_3, x_4, x_5) = 1 \oplus x_2 \oplus x_3 \oplus x_4 x_5 \oplus x_1 x_2 x_3 x_5.$$

In acest caz perioada va fi $(2^{L_1} - 1)(2^{L_2 - 1}) \dots (2^{L_5 - 1})$.

10 DES

DES inseamna *data encryption standard* si a fost vreme de mai multi ani standardul de securitate stabilit de NIST (National Institute of Standards in Technology, USA). Sistemul reprezinta rezultatul unei colaborari intre Horst Feistel si echipa IBM. Acest sistem de criptare nu a fost niciodata spart. Totusi, odata cu cresterea puterii de calcul, a aparut temerea ca blocul de 64 de biti si cheia de 56 de biti sunt prea scurte pentru un sistem sigur. Urmarea a fost un concurs pentru stabilirea unui nou standard de securitate. El a fost castigat de AES, tema paragrafului urmator.

Definitie: O runda Feistel este functia $f : \{0,1\}^{2n} \to \{0,1\}^{2n}$ definita in modul urmator. Fie $F : \{0,1\}^n \to \{0,1\}^n$ o functie oarecare. Impartim cuvintele $x \in \{0,1\}^{2n}$ in jumatati egale $x = (L, R)$ cu $L, R \in \{0,1\}^n$. Atunci:

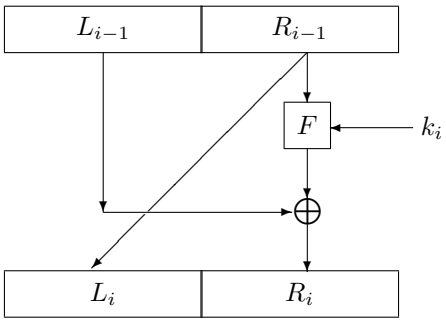
$$f(x) = f(L, R) = (R, L \oplus F(R)).$$

Teorema: Runda Feistel este o functie bijectiva.

Demonstratie: Intr-adevar, F nu trebuie sa fie inversabila. Observam ca:

$$f^{-1}(A, B) = (B \oplus F(A), A).$$

□



Rundele Feistel se cupleaza in serie formand retele Feistel. Functia pseudo-aleatoare F depinde de o cheie de runda k_i. Daca actiunea unei runde Feistel se scrie:

$$\begin{aligned} L_i &= R_{i-1}, \\ R_i &= L_{i-1} \oplus F(k_i, R_{i-1}), \end{aligned}$$

atunci pasul de decriptare se face in modul urmator:

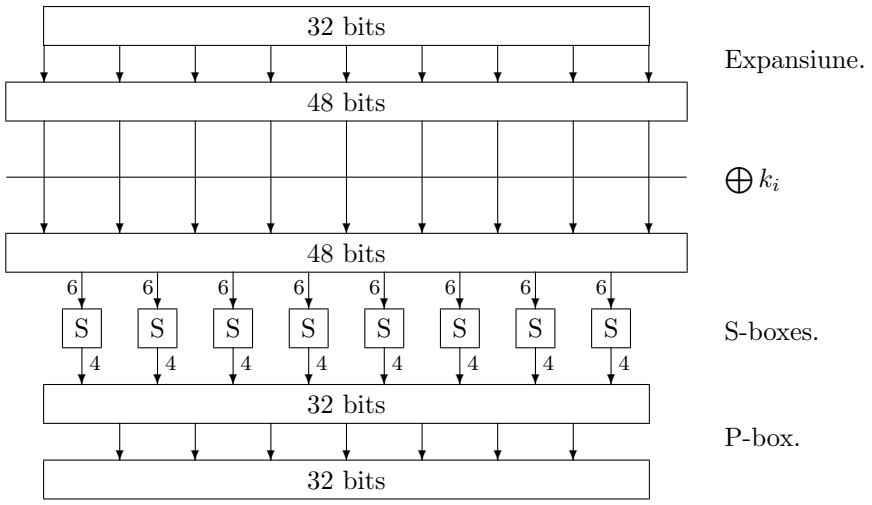
$$\begin{aligned} R_{i-1} &= L_i, \\ L_{i-1} &= F(k_i, L_i) \oplus R_i. \end{aligned}$$

DES este o retea Feistel de 16 runde.

In continuare ne vom ocupa doar de functia F. Ea va fi reprezentata schematic mai jos.

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Permutarea preliminara aplicarii celor 12 runde Feistel. Primele 4 randuri formeaza blocul L, celelalte 4 randuri formeaza blocul R. La sfarsit se aplica inversa acestei permutari.



Schema functiei $F(k_i, \cdot)$ din DES.

Expansiunea. Este o functie $E : \{0,1\}^{32} \to \{0,1\}^{48}$ care actioneaza in felul urmator:

$$f(x_1, x_2, \ldots, x_{32}) = (x_{32}, x_1, x_2, x_3, x_4, x_5, x_4, x_5, \ldots, x_{32}, x_1).$$

Cu alte cuvinte, fiecare pereche de biti x_{4k}, x_{4k+1} se replica in $x_{4k}, x_{4k+1}, x_{4k}, x_{4k+1}$. Aceast proces are loc circular. In final, cei 48 de biti sunt cei 32 de biti initiali plus 16 biti copiati.

Blocul de 48 de biti se aduna binar cu cheia de runda. Fiecare cheie de runda se obtine din cheia principala folosind o functie de key scheduling: $k_i = s(k, i)$. Rezultatul se imparte in blocuri de cate 6 biti. Fiecare bloc de 6 biti este prelucrat de o S-box.

S-box. Fiecare S-box este o functie $S_i : \{0,1\}^6 \to \{0,1\}^4$. Sunt exact 8 S-boxes S_1, ..., S_8. Aceste functii sunt complet definite si standardizate.

Iata de exemplu S_1. Pe coloana din stanga sunt numerele codificate de cei 2 bitii extremi, iar pe prima linie sunt numerele codificate de cei 4 biti intermediari. Astfel pentru a cauta $S_1(010101)$ privim pe linia $01_2 = 1_{10}$ si citim coloana $1010_2 = 10_{10}$. Aflam ca $S_1(010101) = 12_{10} = 1100$.

Toate S-boxes sunt construite astfel incat fiecare linie este o permutare a numerelor 0, ..., 15. Toate au proprietatea ca daca se modifica un bit la intrare, asta duce la modificarea a cel putin doi biti la iesire. Acest comportament corespunde principiului **confuziei** definit de Shannon: modificare radicala a sub-blocurilor.

S_1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S-box S_1 ca lista de rezultate.

P-box. Permutare finala din functia F are proprietatea ca bitii produsi de fiecare S-box sunt transportati in dreptul a 4 alte S-boxes. Aceasta corespunde principiului **difuziei** definit de Shannon: amestecul extrem al rezultatului operatiei de confuzie. Iata mai jos permutarea produsa de P-box afisata ca lista de rezultate:

$$\begin{pmatrix} 16 & 7 & 20 & 21 \\ 29 & 12 & 28 & 17 \\ 1 & 15 & 23 & 26 \\ 5 & 18 & 31 & 10 \\ 2 & 8 & 24 & 14 \\ 32 & 27 & 3 & 9 \\ 19 & 13 & 30 & 6 \\ 22 & 11 & 4 & 25 \end{pmatrix}$$

Un posibil mod de aplicare a algoritmului DES este algoritmul 3DES sau Triple DES. O cheie este concatenarea a trei chei DES. Criptarea se face cu:

$$c = \mathrm{DES}_{k_3}\mathrm{DES}_{k_2}^{-1}\mathrm{DES}_{k_1}(m),$$

iar decriptarea este operatia inversa corespunzatoare.

11 AES

Algoritmul Advanced Encryption Standard i-a urmat lui DES ca standard NIST in octombrie 2000. Autorii lui sunt belgienii Joan Daemen si Vincent Rijmen, fapt pentru care algoritmul se mai numeste si Rijndael.

Algoritmul nu poate fi descris fara niste definitii de reprezentare a numerelor si polinoamelor.

Definitie: In scrierea hexadecimala, adica in baza 16, cifrelor 0, 1, 2, ..., 9, li se adauga cifrele A, B, C, D, E, F. Un numar hexadecimal este intotdeauna reprezentat cu prefixul $0x$. De exemplu:

$$0x83 = 8 \cdot 16 + 3 = 131 = 128 + 2 + 1 = 2^7 + 2 + 1 = X^7 + X + 1.$$

Un alt mod de operare este sa impartim stringurile binare in grupuri de cate 4 care sunt traduse in cifre hexadecimale:

$$X^7 + X + 1 = 10000011 = 1000\ 0011 = 0x83.$$

Definitie: Aritmetica pe corpul $\mathbb{F}_{2^8} = \mathbb{F}_{256}$ este definita de polinomul ireductibil:

$$X^8 + X^4 + X^3 + X + 1.$$

Definitie: Cuvintele de 32 de biti se identifica cu polinomame de grad ≤ 3 din $\mathbb{F}_{256}[X]$. Astfel, un pachet $a_0a_1a_2a_3$ cu $a_i \in 2^8$ se identifica cu polinomul:

$$a_3X^3 + a_2X^2 + a_1X + a_0.$$

Aritmetica pe $\mathbb{F}_{2^8}[X]$ se face modulo X^4+1. Observam ca $X^4+1 = (X^2+1)^2 = (X+1)^4$. Din acest motiv, inelul nu este un corp intrucat are divizori ai lui 0. Intr-adevar, $(X+1)(X^3+X^2+X+1) = 0$.

AES lucreaza cu blocuri de 128 de biti folosind chei de runda de 128 de biti.

Definitie: Starea interna a algoritmului este blocul din registrul principal la un moment dat. Ea se reprezinta ca o matrice de 4×4 octeti (bytes):

$$S = \begin{pmatrix} s_{00} & \cdots & s_{03} \\ \vdots & \ddots & \vdots \\ s_{30} & \cdots & s_{33} \end{pmatrix}$$

Cheia de runda are o forma asemanatoare:

$$K_i = \begin{pmatrix} k_{00} & \dots & k_{03} \\ \vdots & \ddots & \vdots \\ k_{30} & \dots & k_{33} \end{pmatrix}$$

Pentru a prezenta algoritmul, trebuie intai sa definim operatiile elementare:

Sub-Bytes sau **S-box**. Aceasta este singura operatie de **confuzie** si se aplica fiecarui byte in parte. Daca fiecare byte este considerat intai element in $\mathbb{F}_{256}$, aceasta operatie se reprezinta ca $x \rightsquigarrow Ax^{-1} + b$. Mai exact, operatia $x \rightsquigarrow x^{-1}$ este inversul multiplicativ pentru $x \neq 0$ si este $0 \rightsquigarrow 0$ altfel. Partea a doua este operatia $y \rightsquigarrow z = Ay + b$ si se refera la o transformare liniara in spatiul vectorial $\mathbb{F}_2^8$ dupa cum urmeaza:

$$\begin{pmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \\ z_6 \\ z_7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}.$$

Operatia Sub-Bytes poate fi facuta o singura data si stocata intr-o lista sau dictionar, de unde este citita. Acelasi lucru se poate spune si despre inversa acestei permutari a multimii $\{0, \dots, 255\}$.

Acum urmeaza o operatie de difuzie:

Shift Rows. Este un shift ciclic al matricii de stare:

$$\begin{pmatrix} a_{01} & a_{02} & a_{03} & a_{04} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{pmatrix} \rightsquigarrow \begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{11} & a_{12} & a_{13} & a_{10} \\ a_{22} & a_{23} & a_{20} & a_{21} \\ a_{33} & a_{30} & a_{31} & a_{32} \end{pmatrix}$$

Mix Columns. Daca asimilam fiecare coloana a matricii de stare cu un polinom de grad ≤ 3 cu coeficienti in $\mathbb{F}_{256}$, atunci operatia Mix Columns, care este deopotriva o operatie de confuzie si de difuzie, se poate reprezenta in modul urmator:

$$b_0 + b_1X + b_2X^2 + b_3X^3 = (a_0 + a_1X + a_2X^2 + a_3X^3)(2 + X + X^2 + 3X^3) \bmod (X^4 + 1),$$

unde coeficientii 2 si 3 din formula au semnificatia ω si $1 + \omega$ cu interpretarea data de polinomul ireductibil de grad 8 cu ajutorul caruia am definit aritmetica lui $\mathbb{F}_{256}$.

Este mai simplu sa ne imaginam aceasta operatie pentru coloane ale matricii de stare actionand in modul urmator:

$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix}$$

Cum aceasta matrice este inversabila in inelul $\mathcal{M}_{4\times4}(\mathbb{F}_{256})$, operatia inversa se realizeaza tot prin inmultirea cu o matrice.

Add Round Key. Aceasta este operatia $S \rightsquigarrow S \oplus K_i$.

Calcularea cheilor de runda. Fie k cheia principala de 128 de biti. O putem imparti in patru blocuri egale, $k = (k_0, k_1, k_2, k_3)$. Fie $RC_i = X^i \bmod (X^8 + X^4 + X^3 + X + 1)$ constanta de runda. In algoritmul urmator obtinem cheile de runda in formatul $k_i = (w_{4i}, w_{4i+1}, w_{4i+2}, w_{41+3})$.

Definim **Rot Bytes** rotirea unui cuvant catre stanga, cu un byte. Sub Bytes, care a fost definita anterior, se aplica fiecarui byte dintr-un cuvant.

$w_0 = k_0$; $w_1 = k_1$; $w_2 = k_2$; $w_3 = k_3$;
for $i = 1$ **to** 10 **do**
    $T =$ SubBytes RotBytes w_{4i-1};
    $T = T \oplus RC_i$;
    $w_{4i} = w_{4i-4} \oplus T$;
    $w_{4i+1} = w_{4i-3} \oplus w_{4i}$;
    $w_{4i+2} = w_{4i-2} \oplus w_{4i+1}$;
    $w_{4i+3} = w_{4i-1} \oplus w_{4i+2}$;
end;

AES. Avem toate elementele necesare pentru prezentarea algoritmului de criptare:

$S = S \oplus k_0$;
for $i = 1$ **to** 9 **do**
    $S =$ MixColumns ShiftRows SubBytes S;
    $S = S \oplus k_i$;
end;
$S =$ ShiftRows SubBytes S;
$S = S \oplus k_{10}$;

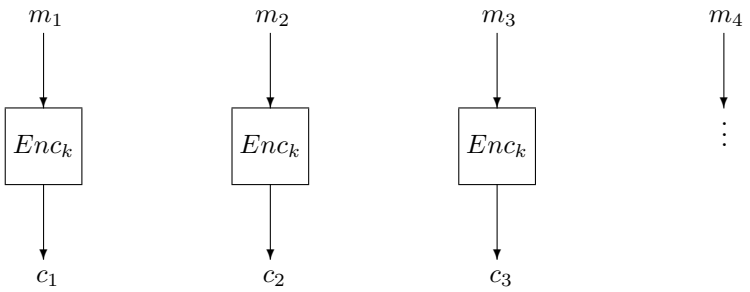
12 Moduri de operare

Fie A un alfabet finit si $Enc : K \times A^n \to A^n$ un sistem de criptare pentru blocuri de lungime n. Se pune problema in ce mod putem aplica sistemul de criptare Enc pentru a cripta (si decripta) mesaje de lungime arbitrara. In practica criptografica s-au cristalizat 5 moduri clasice de aplicare a criptarilor bloc. Fiecare dintre ele are anumite avantaje si dezavantaje, pe care le vom descrie succint.

Principul general este ca mesajul clar m este impartit in blocuri $m_1||m_2||m_3 \ldots$, toate de lungime n. Daca ultimul bloc nu are lungime n, el va fi completat cu blancuri sau cu alt caracter conventional. Blocurile criptate vor fi notate c_1, c_2, c_3, ..., iar mesajul criptat $c = c_1||c_2||c_3|| \ldots$.

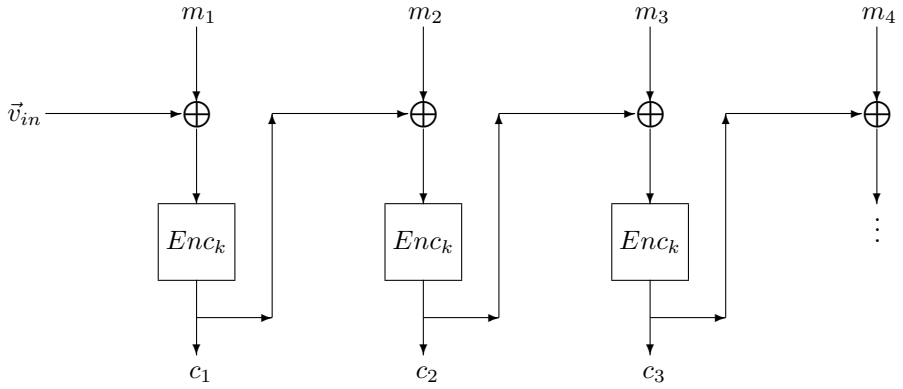
ECB Mode. Electronic Code-book Mode. Fiecare bloc este criptat separat, toate cu aceeasi cheie. Se preteaza la calcul paralel, care este mai rapid. O greseala corupe doar blocul curent. Insa nu reprezinta un nivel ridicat de securitate. Criptarea si decriptarea au loc in modul evident:

$$\begin{aligned} c_i &= Enc_k(m_i), \\ m_i &= Dec_k(c_i). \end{aligned}$$



Modul ECB.

CBC Mode. Cypher Blockchaining Mode. Primul bloc, inainte de criptare, se aduna binar cu un cuvant aleator numit **vector de initializare**. Apoi fiecare bloc, inainte de criptare, se aduna binar cu rezultatul criptarii blocului anterior. Exista si un dezavantaj: daca un bloc din mesajul criptat este corupt, el va corupe decriptarea toturor blocurilor care ii urmeaza.



Modul CBC.

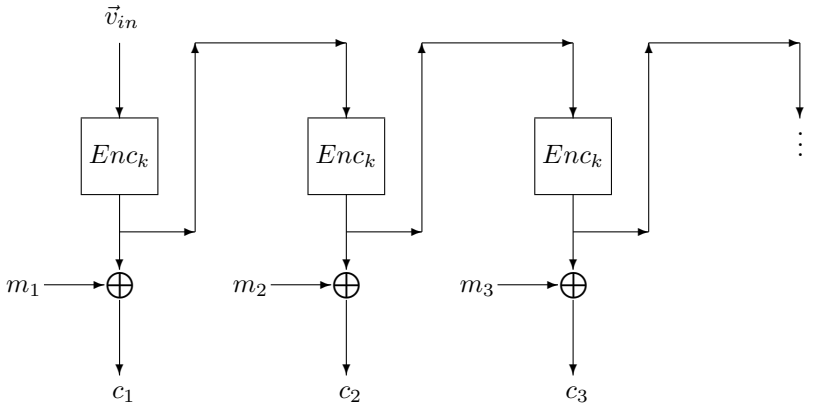
Asadar criptarea este:

$$\begin{aligned} c_1 &= Enc_k(m_1 \oplus \vec{v}_{in}), \\ c_i &= Enc_k(m_i \oplus c_{i-1}), \end{aligned}$$

unde $i \geq 2$, iar decriptarea este:

$$\begin{aligned} m_1 &= Dec_k(c_1) \oplus \vec{v}_{in}, \\ m_i &= Dec_k(c_i) \oplus c_{i-1}. \end{aligned}$$

OFB Mode. Output Feedback Mode. In acest mod de aplicare, este criptat doar vectorul de initializare o data, de doua ori, de trei ori, etc. Blocul m_i se cripteaza prin adunare binara cu respectiva criptare succesiva a vectorului de initializare. Cu alte cuvinte, criptarea bloc este folosita doare pe post de generator pseudo-random. Avantajul este ca acest sir pseudo-aleator se poate calcula dinainte si ca nu avem nevoie de functia *Dec* la decriptare. Dezavantajul este din nou ca daca la criptare s-a stricat un bloc din sirul pseudo-aleator, mesajul nu mai poate fi decriptat.



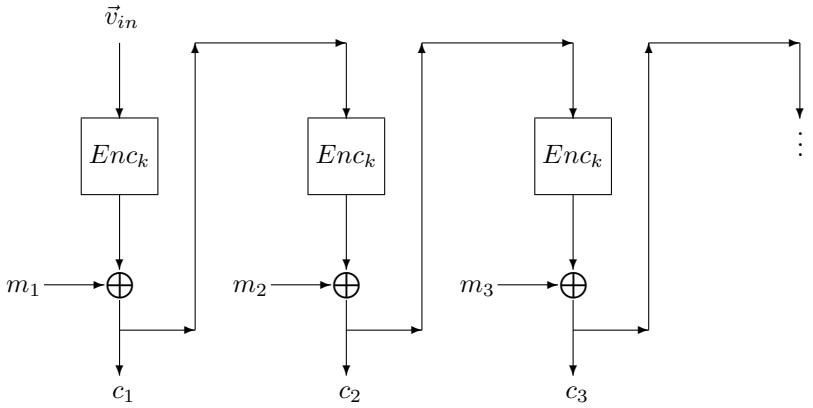
Modul OFB.

Asadar criptarea si decriptarea sunt:

$$\begin{aligned} c_i &= Enc_k^i(\vec{v}_{in}) \oplus m_i, \\ m_i &= Enc_k^i(\vec{v}_{in}) \oplus c_i, \end{aligned}$$

unde $i \geq 1$.

CFB Mode. Cypher Feedback Mode. Acest mod incepe ca si modul OFB. Blocul m_1 este adunat binar cu criptarea unui vector de initializare, si se obtine primul bloc cifrat c_1. al doilea bloc insa se aduna binar cu criptarea lui c_1 si asa mai departe. CFB este asadar o combinatie de OFB si CBC. Este o metoda sigura de criptare, dar lenta in comparatie cu OFB si ECB. Nici CFB nu foloseste functia Dec pentru decriptare.



Modul CFB.

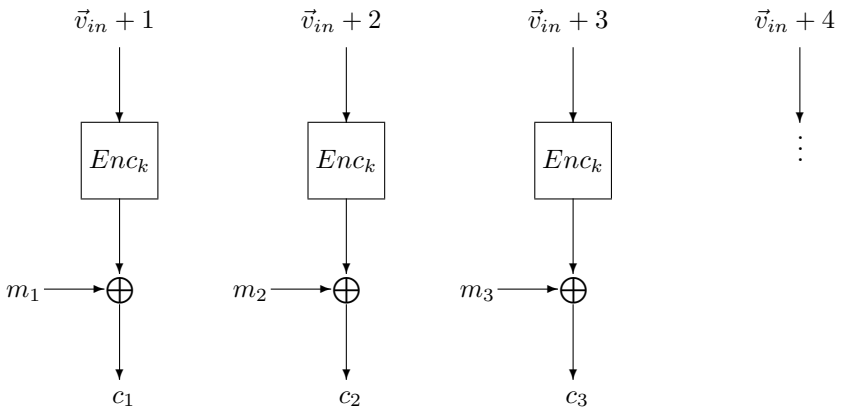
Asadar criptarea este:

$$\begin{aligned} c_1 &= m_1 \oplus Enc_k(\vec{v}_{in}), \\ c_i &= m_i \oplus Enc_k(c_{i-1}), \end{aligned}$$

unde $i \geq 2$, iar decriptarea este:

$$\begin{aligned} m_1 &= c_1 \oplus Enc_k(\vec{v}_{in}), \\ m_i &= c_i \oplus Enc_k(c_{i-1}). \end{aligned}$$

CTR Mode. Counter Mode. In modul CTR vectorul de initializare se modifica de la bloc la bloc intr-un mod previzibil. De exemplu, este tradus in numar natural si la fiecare nou bloc este adunat cu o unitate, apoi retradus in cuvant. Vectorul curent obtinut in acest mod este criptat si adunat binar cu blocul clar curent. Rezultatul este blocul criptat curent. Nici in modul CTR functia *Dec* nu se foloseste la decriptare.



Modul CTR.

Asadar criptarea si decriptarea sunt:

$$\begin{aligned} c_i &= Enc_k(\vec{v}_{in}+i) \oplus m_i, \\ m_i &= Enc_k(\vec{v}_{in}+i) \oplus c_i, \end{aligned}$$

unde $i \geq 1$.

13 Functii hash

In paragraful urmator, prin obiect *greu de gasit* intelegem un obiect pentru a carei calculare avem nevoie de timp exponential $\mathcal{O}(2^n)$, unde n este un parametru de securitate.

Fie A un alfabet. O functie hash $h : A^* \to A^n$ este o functie relativ usor de calculat de la care se asteapta proprietati de genul urmator:

1. *Rezistenta la preimagine.* Dat $y \in A^n$, este greu de gasit un x cu $h(x) = y$.
2. *Rezistenta la a doua preimagine.* Dat $x \in A^*$, este greu de gasit $x' \neq x$ cu $h(x) = h(x')$.
3. *Rezistenta la coliziune.* Este greu de gasit o pereche $x, x' \in A^*$ cu $x \neq x'$ si $h(x) = h(x')$.

Observam ca:

$$\text{RC} \to \text{R2P} \to \text{RP}$$

Alte calitati necesare in aplicatii:

- Cat mai putine coliziuni, deci distributie uniforma a valorilor.

- Haos. Prin asta se intelege inputuri apropiate produc outputuri foarte diferite. La modul ideal, daca inputurile difera printr-un bit, outputurile sa difere in cel putin jumatate din biti.

- Confuzie. Pornind de la valoarea functiei, sa nu ai niciun indiciu despre preimagine.

- Surjectivitate.

- Eficienta.

Constructia Merkle-Damgard. Presupunand ca avem deja o functie $f : \{0,1\}^s \to \{0,1\}^n$ cu $s > n$, rezistenta la coliziuni, se construieste o functie $h : \{0,1\}^* \to \{0,1\}^n$.

input $m = m_1||m_2||\dots||m_t$;
$H = H_0$;
for $i = 1$ **to** t bf do
    $H = f(H||m_i)$;
end;
output H

Ideea de mai sus se poate aplica in modul urmator. Fie $d = s - n$. Atunci se completeaza m cu zerouri, pana cand se obtine o lungime td, multiplu de d. Se imparte inputul completat cu zerouri in t blocuri. Daca H_0 este un sir constant, de lungime n, atunci la fiecare pas functia f primeste un input e lungime $n + d = s$. □

Observatie: Fie $f : \{0,1\}^8 \to \{0,1\}^4$ o functie de compresie rezistenta la coliziuni. Daca aplicam metoda de mai sus pentru $m_1 = 010$ si $m_2 = 0100$ obtinem:

$$h(m_1) = f(0100\ 0000) = h(m_2).$$

Asadar se indica marcarea sfarsitului de mesaj cu un 1. In cazul acesta:

$$\begin{aligned} h(m_1) &= f(0101\ 0000), \\ h(m_2) &= f(0100\ 1000), \end{aligned}$$

care sunt in general diferite.

Pentru a intelege mai bine functiile de compresie, vom prezenta functia **MD4**. Pentru $u, v, w \in \{0,1\}^{32}$ definim:

$$\begin{aligned} f(u,v,w) &= (u \wedge v) \vee (\neg u \wedge v) \\ g(u,v,w) &= (u \wedge v) \vee (u \wedge w) \vee (v \wedge w) \\ h(u,v,w) &= u \oplus v \oplus w \end{aligned}$$

pe componente. Exista o stare curenta (A, B, C, D) cu $A, B, C, D \in \{0,1\}^{32}$. Se dau urmatoarele valori initiale:

$$\begin{aligned} H_1 &= 0x67452301, \\ H_2 &= 0xEFCDAB89, \\ H_3 &= 0x98BADCFE, \\ H_4 &= 0x10325476, \end{aligned}$$

si constante (y_j, z_j, s_j) care depind de fiecare runda j. Inputul este impartit in blocuri:

$$X = X_0||X_1||\dots||X_{15},$$

unde fiecare bloc are 32 biti. Asadar $|X| = 16 \times 32 = 512$. La primul pas starea interna se initializeaza cu valoarea initiala:

$$(A, B, C, D) = (H_1, H_2, H_3, H_4)$$

Algoritmul consta din 3 pachete de cate 16 runde. Primul pachet este:

for $j = 0$ **to** 15 **do**

$t = A \oplus f(B,C,D) \oplus X_{z_j} \oplus y_j$;

$(A,B,C,D) = (D, t <<< s_j, B, C)$

end;

Al doilea pachet, cu $j = 16$ to 31, este identic, dar foloseste functia g in loc de f. Al treilea pachet, cu $j = 32$ to 47 foloseste funtia h in loc de g. La sfarsit se calculeaza:

$$(A,B,C,D) = (A,B,C,D) \oplus (H_1, H_2, H_3, H_4).$$

Asadar functia MD4 : $\{0,1\}^{512} \to \{0,1\}^{128}$ este calculata tot printr-o succesiune de confuzii si de difuzii. □

Merkle-Damgard nu este singura metoda de constructie a unei functii hash generale. Functii hash se pot obtine si folosind **criptari bloc**, cu conditia ca ele sa respecte normele de confuzie si de difuzie. Aceste metode sunt numite generic *one-way compression functions*. Amintim aici cateva metode:

Matyas-Meyer-Oseas

$$H_i = f(x_i, H_{i-1}) = Enc_{H_{i-1}}(x_i) \oplus x_i$$

Davies-Meyer

$$H_i = f(x_i, H_{i-1}) = Enc_{x_i}(H_{i-1}) \oplus H_{i-1}$$

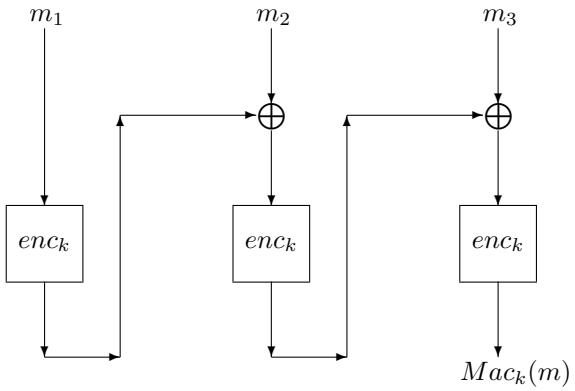
Miyaguchi-Preenel

$$H_i = f(x_i, H_{i-1}) = Enc_{H_{i-1}}(x_i) \oplus x_i \oplus H_{i-1}$$

Una din principalele aplicatii ale functiilor hash sunt protocoalele de autentificare a mesajelor, *Message Authentification Codes*. In cadrul acestor protocoale, mesajul este trimis impreuna cu valoarea unei functii hash. Cum functia poate fi calculata doar daca esti in posesia unei chei de autentificare, aceasta este o masura suplimentara de securitate. Ea este aplicata in protocoale automate, in care receiverul nu mai descifreaza mesajul daca el nu este autentic. Cea mai buna varianta de autentificare are forma:

$$Enc_{k_1}(m) || Mac_{k_2}(Enc_{k_1}(m)).$$

Aici este important ca Mac sa fie alt algoritm decat Enc iar cheile de criptare si de autentificare sa fie diferite. Pentru functia Mac se poate folosi orice functie hash performanta, sau urmatoarea varianta numita CBC-Mac. Aici este important ca functia *enc* sa fie alta functie decat functia Enc folosita pentru criptare. Exemplu pentru 3 blocuri:



CBC-Mac

Part III

Criptografie cu cheie publica

14 Cheie publica si cheie secreta

Conceptul de criptografie cu cheie publica provine din observatia ca nu toate perechile de partcipanti la comunicatie sunt simetrice. In multe situatii o autoritate (institutie de stat, banca, etc) este contactata de multi utilizatori, iar in cadrul comunicarii cele doua parti nu au un rol simetric. De exemplu, institutia de stat primeste cereri si reclamatii la care nu va raspunde imediat. Cetatenii nu vor sa se stie in ce problema s-au adresat institutiei, dar vor ca institutia sa poata descifra mesajul. Daca Ministerul Finantelor ar avea un protocol de criptare simetrica cu fiecare cetatean, ar trebui sa administreze 22.000.000 de chei. Acest sistem ar fi foarte costisitor si foarte nesigur.

Ideea este sa se inlocuiasca cele 22.000.000 de chei simetrice (k_i) cu o singura pereche de chei (p, s). Cheia publica p este cunoscuta de toata lumea. Oricine poate calcula un mesaj criptat:

$$c = Enc_p(m),$$

si il poate trimite ministerului pe un canal public. Cheia secreta (privata) s este cunoscuta numai Ministerului Finantelor. Ministerul poate calcula mesajul clar:

$$m = Dec_s(c),$$

dar nimeni altcineva nu poate calcula mesajul clar fara cunoasterea acestei chei. De asemeni, nici cunoasterea cheii publice p si nici a unor mesaje criptate, sau observarea procesului de criptare cu mesaje alese, nu te ajuta la aflarea cheii secrete s.

Fenomenul fundamental care sta la baza criptografiei cu cheie publica este **functia one-way**. Acestea sunt functii usor de calculat ca functie directa, dar greu de calculat ca functie inversa. Un exemplu clasic de functie one-way este factorizarea. Daca produsul a doua numere prime:

$$(p, q) \to pq,$$

este usor de efectuat, este mult mai greu de factorizat un asemenea produs:

$$pq \to (p, q).$$

15 Probleme matematice

In aceasta sectiune prezentam o lista de probleme matematice pe a caror presupusa dificultate se bazeaza securitatea criptografiei cu cheie publica. Mai jos p si q sunt numere prime necunoscute, $p \neq q$, iar $N = pq$ este cunoscut.

FACTORING. Dat $N = pq$, produs de doua numere prime diferite, sa se afle p.

RSA. Dat e astfel incat $\gcd(e, (p-1)(q-1)) = 1$ si dat c, sa se afle m astfel incat $m^e = c \bmod N$.

QUADRES. Dat a, sa se determine daca a este un patrat modulo N.

SQRROOT. Dat a astfel incat $a = x^2 \bmod N$, sa se gaseasca un x care satisface relatia.

Daca FACTORING se rezolva determinist in timp polinomial, atunci acelasi lucru s-ar spune si despre RSA, QUADRES si SQRROOT. Nu este evident, dar vom argumenta in paragrafele urmatoare.

DLP. Dat $(G, \cdot)$ un grup abelian finit, dati $g, h \in G$ astfel incat $h \in \langle g \rangle$, sa se afle x astfel incat $h = g^x$. Aceasta problema se numeste *Logaritm Discret*.

DHP. Dat $(G, \cdot)$ un grup abelian finit, $g \in G$, si doua elemente $a, b \in G$ astfel incat $a = g^x$ si $b = g^y$, sa se afle un $c \in G$ astfel incat $c = g^{xy}$. Aceasta problema se numeste *Diffie-Hellman*.

DDH. Dat $(G, \cdot)$ grup abelian finit, $g \in G$, $a = b^x$, $b = g^y$ si $c = g^z$, sa se determina daca $z = xy$.

Daca DLP se rezolva determinist in timp polinomial, atunci acelasi lucru s-ar spune si despre DHP si DDH.

Deocamdata pentru niciuna din aceste probleme nu se cunosc algoritmi deterministi in timp polinomial. Pe de alta parte se stie ca problemele fac parte din clasa NP, dar pentru niciuna nu se stie ca ar fi NP-completa.

16 Patrate in corpuri prime

Fie aplicatia $sq : \mathbb{F}_p \to \mathbb{F}_p$ data de $sq(x) = x^2$. Se observa ca $sq : \mathbb{F}_p^\times \to \mathbb{F}_p^\times$ este homomorfism de grupuri. Cum $sq(x) = sq(-x)$, $sq(\mathbb{F}_p^\times) \leq \mathbb{F}_p^\times$ cu indice $\mathbb{F}_p^\times : sq(\mathbb{F}_p^\times)] = 2$. Deci exact $(p-1)/2$ elemente sunt patrate, pentru p prim impar.

Definitie: Fie p un numar prim. Simbolul lui Legendre este functia:

$$\left(\frac{\cdot}{p}\right) : \mathbb{Z} \to \{-1, 0, 1\},$$

$$\left(\frac{a}{p}\right) = \begin{cases} 0 & \text{daca } p|a, \\ +1 & \text{daca } a \bmod p \in sq(\mathbb{F}_p^\times), \\ -1 & \text{altfel.} \end{cases}$$

Se observa ca:

$$\left(\frac{a}{p}\right) = a^{\frac{p-1}{2}} \bmod p,$$

eficienta cu exponentiere rapida. Pentru numere compuse se folosesc urmatoarele criterii:

Legea de Reciprocitate Patratica a lui Gauss. Daca p si q sunt numere prime impare, atunci:

$$\left(\frac{q}{p}\right) = \left(\frac{p}{q}\right)(-1)^{\frac{(p-1)(q-1)}{4}},$$

explicit:

$$\left(\frac{q}{p}\right) = \begin{cases} -\left(\frac{p}{q}\right) & p \bmod 4 = q \bmod 4 = 3, \\ +\left(\frac{p}{q}\right) & \text{altfel.} \end{cases}$$

Se mai adauga urmatoarele reguli de calcul:

$$\begin{aligned} \left(\frac{q}{p}\right) &= \left(\frac{q \bmod p}{p}\right), \\ \left(\frac{qr}{p}\right) &= \left(\frac{q}{p}\right)\left(\frac{r}{p}\right), \\ \left(\frac{2}{p}\right) &= (-1)^{\frac{p^2-1}{8}}. \end{aligned}$$

Observatie: Daca FACTORING se poate face in timp polinomial, atunci si QUADRES. Exemplu:

$$\left(\frac{15}{17}\right) = \left(\frac{3}{17}\right)\left(\frac{5}{17}\right) = \left(\frac{17}{3}\right)\left(\frac{17}{5}\right) = \left(\frac{2}{3}\right)\left(\frac{2}{5}\right) = (-1)(-1)^3 = 1.$$

Observatie: QUADRES este mai usoara decat SQRROOT. Intr-adevar, daca avem un algoritm rapid de calcul, el este in acelasi timp un algoritm de decizie.

Vom arata ca pentru $N = pq$, FACTORING si SQRROOT au practic aceeasi dificultate, relativ la timpul polinomial. Prima observatie este ca exista un algoritm eficient pentru a extrage radacina patrata modulo numar prim.

Algoritmul lui Shanks de extragere a radacinei patrate modulo p.

Inputs: p prim, $n \in \mathbb{Z}_p^\times$.

Find $Q = 1 \bmod 2$, S: $p - 1 = 2^S Q$.

Find $z \in \mathbb{Z}_p^\times$: $\left(\frac{z}{p}\right) = -1$.

$M = S$.

$c = z^Q$.

$t = n^Q$.

$R = n^{\frac{Q+1}{2}}$.

Loop: **if** $t = 1$ **return** R.

Find least i: $0 < i < M \wedge t^{2^i} = 1$ **or return NO**.

$b = c^{2^{M-i-1}}$.

$M = i$.

$c = b^2$.

$t = tb^2$.

$R = Rb$.

Teorema: *Algoritmul lui Shanks calculeaza eficient radacina patrata modulo numar prim.*

Demonstratie: Observam ca in timpul aplicarii algoritmului, urmatoarele cantitati raman constante:

$$\begin{aligned} c^{2^{M-1}} &= -1, \\ t^{2^{M-1}} &= 1, \\ R^2 &= tn. \end{aligned}$$

La inceput:

$$c^{2^{M-1}} = z^{2^{S-1}Q} = z^{\frac{p-1}{2}} = -1,$$

deoarece z nu este rest patratic,

$$t^{2^{M-1}} = n^{2^{S-1}Q} = n^{\frac{p-1}{2}} = 1,$$

deoarece n este rest patratic,

$$R^2 = n^{Q+1} = nt.$$

La fiecare noua iteratie, fie (M', c', t', R') noile valori ale tuplului (M, c, t, R).

$$c'^{2^{M'-1}} = (b^2)^{2^{i-1}} = c^{2^{M-i} \cdot 2^{i-1}} = c^{2^{M-1}} = -1.$$

$$t'^{2^{M'-1}} = (tb^2)^{2^{i-1}} = t^{2^{i-1}} b^{2^i} = (-1)(-1) = 1,$$

unde $t^{2^{i-1}} = -1$ deoarece $t^{2^i} = 1$ si $t^{2^{i-1}} \neq 1$, altfel nu ar fi acest i cea mai mica valoare.

$$b'^{2^i} = c^{2^{M-i-1} \cdot 2^i} = c^{2^{M-1}} = -1,$$

$$R'^2 = R^2 b^2 = tnb^2 = t'n.$$

Dar M se micsoreaza la fiecare iterare, deci in final t va fi egal cu 1, caz in care $R^2 = n$. Toate calculele sunt modulo p. □

Exemplu: Fie $p = 17$ si $n = 15$. Toate calculele au loc modulo 17. Dupa cum am vazut,

$$\left(\frac{15}{17}\right) = 1,$$

deci are rost sa cautam radacina patrata.

$$p - 1 = 16 = 2^4 \cdot 1,$$

deci $Q = 1$ si $S = 4$. Numarul $z = 3$ nu este un rest patratic, pentru ca:

$$3^8 = 9 \cdot 27^2 = 9 \cdot 10^2 = 9 \cdot 4 \cdot 25 = 9 \cdot 4 \cdot 8 = 18 \cdot 16 = 1 \cdot (-1) = -1.$$

Asadar pornim cu:

$$\begin{aligned} M = S &= 4 \\ c = z^Q &= 3 \\ t = n^Q &= 15 \\ R = n^{\frac{1+1}{2}} &= 15 \end{aligned}$$

Prin ridicari succesive la patrat gasim cel mai mic $i < 4$ astfel incat $15^{2^i} = 1$.

$$15^2 = (-2)^2 = 4 \rightsquigarrow 4^2 = 16 = -1 \rightsquigarrow (-1)^2 = 1.$$

Deci $i = 3$.

$$b = c^{2^{M-i-1}} = 3^{2^{4-3-1}} = 3,$$

$$\begin{aligned} M = i &= 3 \\ c = b^2 &= 9 \\ t = tb^2 = 15 \cdot 9 &= 16 \\ R = Rb = 15 \cdot 3 &= 11 \end{aligned}$$

Prin ridicari succesive la patrat gasim cel mai mic $i < 3$ astfel incat $16^{2^i} = 1$.

$$16^2 = (-1)^2 = 1.$$

Deci $i = 1$.

$$b = c^{2^{M-i-1}} = 9^{2^{3-1-1}} = 3 \cdot 27 = 30 = 13 = -4,$$

$$\begin{aligned} M = i &= 1 \\ c = b^2 &= 16 \\ t = tb^2 = 16 \cdot 16 &= 1 \\ R = Rb = 11 \cdot (-4) &= -10 = 7 \end{aligned}$$

Cum $t = 1$ se da rezultatul 7. Intr-adevar, $49 \bmod 17 = 15$, OK. Cealalta solutie este $-7 = 10$.

Observatie: Numarul maxim de runde in algoritmul lui Shanks este exponentul lui 2 in $p - 1$. Acest numar este $< \log p$, deci algoritmul este in timp polinomial.

Teorema: *FACTORING si SQRROOT sunt polinomial-time echivalente pentru numerele $N = pq$, produse de doua numere prime diferite.*

Demonstratie: Presupunem ca putem rezolva FACTORING in timp polinomial. Asadar gasim numerele prime p si q. Cu algoritmul lui Shanks gasim s_1 si s_2 astfel incat:

$$\begin{aligned} s_1 &= \sqrt{n} \bmod p, \\ s_2 &= \sqrt{n} \bmod q. \end{aligned}$$

Cu Teorema Chineza a resturilor se gaseste un x cu $0 \leq x < pq$ astfel incat:

$$\begin{aligned} s_1 &= x \bmod p, \\ s_2 &= x \bmod q. \end{aligned}$$

Deci:

$$\begin{aligned} x^2 \bmod p &= n, \\ x^2 \bmod q &= n, \end{aligned}$$

de unde $x^2 \bmod pq = n$ deoarece p si q sunt prime intre ele.

Acum sa presupunem ca putem rezolva SQRROOT in timp polinomial pentru $N = pq$ cu $p \neq q$ prime. Alegem la intamplare $x \in \mathbb{Z}_N^\times$. Fie $x = z^2 \bmod N$ si $y = \sqrt{z} \bmod N$. Exista 4 asemenea radacini, deoarece $N = pq$. Cu probabilitate de 50% obtinem $y \neq \pm x \bmod N$, altfel repetam procedura. Din $x^2 = y^2 \bmod N$ rezulta $N|(x+y)(x-y)$. Dar cum $N \not| (x+y)$ si $N \not| (x-y)$ rezulta ca factorii lui N sunt distribuiti aici. Deci $\gcd(N, x-y)$ este p sau q. □

17 Algoritmul lui Cipolla

Algoritmul lui Cipolla pentru extragerea radacinii patrate modulo p este mai usor de memorat decat algoritmul lui Shanks, si se preteaza mai bine pentru exercitii. Iata algoritmul:

Input $n \in sq(\mathbb{F}_p^\times)$.

Find $a \in \mathbb{F}_p^\times$ **such that** $a^2 - n \notin sq(\mathbb{F}_p^\times)$.

$\omega = \sqrt{(a^2 - n)} \notin \mathbb{F}_p$

$x = (\omega + a)^{\frac{p+1}{2}}$.

Output x.

Teorema: *Algoritmul lui Cipolla produce un $x \in \mathbb{F}_p$ cu $x^2 = n$.*

Demonstratie: Calculul are loc in corpul $\mathbb{F}_{[\omega]} \simeq \mathbb{F}_{p^2}$ construit cu regula $\omega^2 = a^2 - n$, care nu e un patrat. Cum:

$$\omega^{p-1} = (\omega^2)^{\frac{p-1}{2}} = -1,$$

are loc $\omega^p = -\omega$. Asadar pentru $x, y \in \mathbb{F}_p$,

$$(x + \omega y)^p = (x - \omega y).$$

Deci:

$$x^2 = (a+\omega)^{p+1} = (a+\omega)(a+\omega)^p = (a+\omega)(a-\omega) = a^2 - \omega^2 = a^2 - \big(a^2 - n\big) = n.$$

Cum stim insa ca polinomul $X^2 - n$ are doua solutii x si $p - x$ in $\mathbb{F}_p$, rezulta ca solutia calculata x este in $\mathbb{F}_p$. □

Exemplu: Sa calculam din nou $\sqrt{15}$ in $\mathbb{F}_{17}$. Alegem $a = 1$ si $a^2 - n = 1 - 15 = 3 \bmod 17$. Numarul 3 nu este un patrat, deoarece sirul 3^n este:

$$3, 9, 10, 13, 5, 15, 11, -1,$$

deci $3^8 = -1$. Daca ar fi fost un patrat, am fi avut $3^8 = z^{16} = 1$, pentru ca $\mathbb{Z}_{17}^\times$ are 16 elemente. Deci $\omega^2 = 3$. Trebuie sa calculam:

$$x = (1+\omega)^9.$$

Cum $9 = 8 + 1$,

$$\begin{aligned} (1+\omega)^2 &= 4 + 2\omega, \\ (1+\omega)^4 &= 11 - \omega, \\ (1+\omega)^8 &= 5 - 5\omega, \\ (1+\omega)^9 &= (1+\omega)(5-5\omega) = 5(1-3) = 7. \end{aligned}$$

Intr-adevar $x = 7$ si $x = 17 - 7 = 10$ sunt cele doua solutii. □

18 RSA

RSA, numit dupa creatorii lui Rivest, Shamir, Adleman, este cel mai vechi algoritm de criptare cu cheie publica inca in uz. Vom prezenta initial varianta publicata si demonstratia originala. Intr-o a doua etapa vom face o analiza algebrica mai profunda. Algoritmul va fi prezentat ca dialog intre doua parti: autoritatea Alice si clientul Bob.

Setup.

Alice:

1. Alege numere prime mari $p \neq q$.
2. Calculeaza $N = pq$ si $\varphi(N) = pq(1 - \frac{1}{p})(1 - \frac{1}{q}) = (p-1)(q-1) = |\mathbb{Z}_N^\times|$.
3. Alege e astfel incat $\gcd(e, \varphi(N)) = 1$.
4. Anunta cheia publica (N, e).
5. Gaseste d astfel incat $d = e^{-1} \bmod \varphi(N)$.
6. Pastreaza cheia secreta d.

Protocol.

Bob: Are de criptat un mesaj $m \in \mathbb{Z}_N$. Trimite $c = m^e \bmod N$.

Alice: Calculeaza $m = c^d \bmod N$.

Teorema: *Daca $p \neq q$ sunt numere prime, $N = pq$, $\varphi(N) = (p-1)(q-1)$, $\gcd(e, \varphi(N)) = 1$, $d = e^{-1} \bmod \varphi(N)$ si $0 \leq m < N$, atunci:*

$$(m^e \bmod N)^d \bmod N = m.$$

Demonstratie: Deoarece $ed = 1 \bmod (p-1)(q-1)$, rezulta ca:

$$\begin{aligned} ed &= 1 \bmod (p-1), \\ ed &= 1 \bmod (q-1), \end{aligned}$$

adica pentru anumiti $k, h \in \mathbb{N}$,

$$\begin{aligned} ed &= k(p-1) + 1, \\ ed &= h(q-1) + 1. \end{aligned}$$

Observam ca:

$$(m^e \bmod N)^d \bmod N = m^{ed} \bmod N.$$

Daca $\gcd(m, p) = 1$ atunci $m^{p-1} = 1 \bmod p$ deoarece $\mathbb{Z}_p$ este un corp. Deci:

$$m^{ed} = m^{k(p-1)+1} = (m^{p-1})^k m = 1^k m = m \bmod p.$$

Daca $p|m$, atunci:

$$m^{ed} = 0 = m \bmod p.$$

In ambele cazuri, deci:

$$m^{ed} = m \bmod p.$$

In acelasi mod se arata si ca:

$$m^{ed} = m \bmod q.$$

Dar p si q sunt numere prime distincte, asadar $\gcd(p,q)=1$, iar $N=pq$ si rezulta:

$$m^{ed}=m \bmod N.$$

□

Exemplu: $N=91=7\cdot 13$, $\varphi(91)=6\cdot 12=72$. Alegem $e=5$. Pentru calculul lui $d=e^{-1} \bmod 72$, efectuam Euclid extins:

$$\begin{aligned} 72 &= 14\cdot \underline{5}+\underline{2}, \\ \underline{5} &= 2\cdot \underline{2}+1. \end{aligned}$$

$$1=\underline{5}-2\cdot\underline{2}=\underline{5}-2\cdot(-14\cdot\underline{5})=29\cdot 5 \bmod 72,$$

deci $d=29$. Daca $m=10$, $c=10^5=10^4\cdot 10 \bmod 91$. Ridicarea succesiva la patrat da:

$$10\rightsquigarrow 100=9\rightsquigarrow 81,$$

deci $c=81\cdot 10 \bmod 91=-100 \bmod 91=82$. Pentru decriptare, calculam $m=82^{29} \bmod 91=82^{16+8+4+1} \bmod 91$. Prin ridicare succesiva la patrat:

$$82^2=(-9)^2=81\rightsquigarrow 81^2=(-10)^2=9\rightsquigarrow 81\rightsquigarrow 81^2=(-10)^2=9,$$

asadar $82^2=81$, $82^4=9$, $82^8=81$, $82^{16}=9$. In final:

$$82^{16+8+4+1}=9\cdot 81\cdot 9\cdot 82=(-10)(-10)(-9)=(-10)\cdot 90=(-10)\cdot(-1)=10 \bmod 91.$$

□

Exemplu: $N=91=7\cdot 13$, $\varphi(91)=6\cdot 12=72$. Alegem $e=13$. Pentru calculul lui $d=e^{-1} \bmod 72$, efectuam Euclid extins:

$$\begin{aligned} 72 &= 5\cdot \underline{13}+\underline{7}, \\ \underline{13} &= \underline{7}+\underline{6}, \\ \underline{7} &= \underline{6}+1. \end{aligned}$$

$$1=\underline{7}-\underline{6}=\underline{7}-(\underline{13}-\underline{7})=2\cdot\underline{7}-\underline{13}=2\cdot(-5\cdot\underline{13})-\underline{13}=(-11)\cdot 13=61\cdot 13 \bmod 72.$$

Deci $d=61$. Totusi, aici apare o problema neasteptata. Fie $0\le m<91$ un mesaj. Ne reamintim reprezentarea data de Teorema Chineza a Resturilor:

$$m \bmod 91=(m \bmod 7, m \bmod 13).$$

Deci:

$$m^{13} \bmod 91=(m^{6\cdot 2+1} \bmod 7, m^{12+1} \bmod 13)=(m \bmod 7, m \bmod 13)=m \bmod 91.$$

Cu alte cuvinte, cheia $e=13$ nu cripteaza efectiv niciun mesaj, si acelasi lucru se intampla si cu inversa ei, $e=61$. □

Ce se intampla in acest exemplu? Se dovedeste ca abordarea initiala RSA este prea superficiala, desi a fost preluata asa de catre toate textele de criptografie si de catre toate sursele online. Pentru a evita aparitia unor chei moarte ca mai sus si pentru a gasi numai adevaratele chei, ne lansam acum intr-o dezvoltare teoretica care va culmina cu reformularea protocolului RSA.

Definitie: Un numar intreg se numeste liber de patrate, daca nu se divide prin niciun patrat diferit de 1. In descompunerea in factori primi a unui numar liber de patrate, toate numerele prime apar pa puterea 1.

Definitie: Cel mai mic multiplu comun a doua numere $\operatorname{lcm}(a,b)$ se defineste ca:

$$\operatorname{lcm}(a,b)=\frac{ab}{\gcd(a,b)}$$

Definitie: Fie N un numar liber de patrate si $N = p_1 p_2 \dots p_k$ descopunerea lui in factori primi distincti. Definim:

$$\lambda(N) = \text{lcm}(p_1 - 1, p_2 - 1, \dots, p_k - 1).$$

Mica Teorema a lui Fermat Generalizata: *Fie N liber de patrate si fie $e = \lambda(N) + 1$. Atunci pentru orice $x \in \mathbb{Z}$,*

$$x^e = x \bmod N.$$

Demonstratie: Pentru toti $i = 1, \dots, k$, $e = (p_i - 1)f_i + 1$, deci:

$$n^e = (n^{p_i - 1})^{f_i} \cdot n = n \bmod p_i,$$

deoarece daca $n = 0 \bmod p_i$ atunci $n^e = 0 \bmod p_i$, pe cand daca $n \neq 0 \bmod p_i$ atunci $n^{p_i - 1} = 1 \bmod p_i$. Cum numerele p_i sunt toate diferite, aplicam Teorema Chineza a Resturilor. □

O prima observatie este ca daca p este prim impar, $n = n^{p-1} \bmod 2p$. Aceasta este deja un enunt mai puternic decat Mica Teorema a lui Fermat.

Aceasta teorema explica la modul general aparitia cheii moarte in exemplul precedent. Inlocuind $\varphi(N)$ cu $\lambda(N)$ vom reformula protocolul RSA:

Setup.

Alice:

1. Alege numere prime mari $p \neq q$.
2. Calculeaza $N = pq$ si $\lambda(N) = \text{lcm}(p-1)(q-1)$.
3. Alege e astfel incat $\gcd(e, \lambda(N)) = 1$.
4. Anunta cheia publica (N, e).
5. Gaseste d astfel incat $d = e^{-1} \bmod \lambda(N)$.
6. Pastreaza cheia secreta d.

Protocol.

Bob: Are de criptat un mesaj $m \in \mathbb{Z}_N$. Trimite $c = m^e \bmod N$.

Alice: Calculeaza $m = c^d \bmod N$.

Teorema: *Daca $p \neq q$ sunt numere prime, $N = pq$, $\lambda(N) = \text{lcm}(p-1)(q-1)$, $\gcd(e, \lambda(N)) = 1$, $d = e^{-1} \bmod \lambda(N)$ si $0 \leq m < N$, atunci:*

$$(m^e \bmod N)^d \bmod N = m.$$

Demonstratie: Deoarece $ed = 1 \bmod \lambda(N)$, rezulta ca:

$$\begin{aligned} ed &= 1 \bmod (p-1), \\ ed &= 1 \bmod (q-1), \end{aligned}$$

adica pentru anumiti $k, h \in \mathbb{N}$,

$$\begin{aligned} ed &= k(p-1) + 1, \\ ed &= h(q-1) + 1. \end{aligned}$$

Restul demonstratiei este identic cu demonstratia precedenta. □

La prima vedere pierdem o groaza de posibile chei de criptare, adica putem face mai putine criptari. Acest lucru nu este adevarat, asa cum arata urmatoarele teoreme:

Teorema: *Fie* $3 \leq e_1 < \varphi(N)$ *astfel incat* $\gcd(e_1, \varphi(N)) = 1$ *si fie* $e_2 = e_1 \bmod \lambda(N)$. *Atunci pentru orice* $m \in \mathbb{Z}$*:*

$$m^{e_1} = m^{e_2} \bmod N.$$

Demonstratie:

$$m^{e_1} = m^{k\lambda(N)+e_2} = (m^{\lambda(N)})^k m^{e_2} = 1^k m^{e_2} = m^{e_2} \bmod N.$$

□

Asadar numai cheile dintre 3 si $\lambda(N)$ dau criptari diferite, dupa aceea se repeta. Daca cineva foloseste o cheie e intre 3 si $\varphi(N)$, persoana care descifreaza poate folosi cheia $(e \bmod \lambda(N))^{-1} \bmod \lambda(N)$ pentru descifrare. Mai mult, pentru a evita cheile moarte se poate alege cheia de criptare $e' \in [3, \lambda(N) - 1]$ dar se publica o cheie de criptare $e = k\lambda(N) + 1$.

Cu aceasta ocazie subliniez ca NIST prezinta varianta de RSA cu $\lambda(N)$. Este un mister pentru mine de ce aceasta varianta nu si-a croit drumul in cartile de criptografie si in sursele online.

Reluam acum primul exemplu in varianta $\lambda(N)$.

Exemplu: $N = 91 = 7 \cdot 13$, $\lambda(91) = \text{lcm}(6, 12) = 12$. Alegem $e = 5$. Pentru calculul lui $d = e^{-1} \bmod 12$, efectuam Euclid extins:

$$\begin{aligned} 12 &= 2 \cdot \underline{5} + \underline{2}, \\ \underline{5} &= 2 \cdot \underline{2} + 1. \end{aligned}$$

$$1 = \underline{5} - 2 \cdot \underline{2} = \underline{5} - 2 \cdot (-2 \cdot \underline{5}) = 5 \cdot 5 \bmod 12,$$

deci $d = 5$. Daca $m = 10$, $c = 10^5 = 10^4 \cdot 10 \bmod 91$. Ridicarea succesiva la patrat da:

$$10 \rightsquigarrow 100 = 9 \rightsquigarrow 81,$$

deci $c = 81 \cdot 10 \bmod 91 = -100 \bmod 91 = 82$. Pentru decriptare, calculam $m = 82^5 \bmod 91 = 82^{4+1} \bmod 91$. Prin ridicare succesiva la patrat:

$$82^2 = (-9)^2 = 81 \rightsquigarrow 81^2 = (-10)^2 = 9,$$

asadar $82^2 = 81$, $82^4 = 9$. In final:

$$82^{4+1} = 9 \cdot 82 = 9 \cdot (-9) = -81 = 10 \bmod 91.$$

□

Avantajele practice ale variantei mod $\lambda(N)$ sunt evidente.

19 Elgamal

Al doilea criptosistem important cu cheie publica este Elgamal. Spre deosebire de RSA, a carui securitate se bazeaza pe dificultatea problemei FACTORING, Elgamal se bazeaza pe dificultatea DLG, adica a logaritmului discret.

In cele ce urmeaza, multimea mesajelor clare M este asociata de la bun inceput cu actiunea unui grup ciclic $(G, \cdot, 1)$ relativ mare. Adica, exista o actiune a grupului G pe multimea M:

$$\alpha : G \times M \to M,$$

notata $\alpha(g, m) = gm$. Aceasta actiune are urmatoarele proprietati:

$$\begin{aligned} g_1(g_2 m) &= (g_1 g_2)m, \\ 1m &= m, \\ g \neq 1 &\rightarrow gm \neq m. \end{aligned}$$

Fie g generatorul grupului ciclic. Algoritmul abstract are urmatoarea descriere:

Setup.

Alice:

1. Alege la intamplare un numar x. Acest numar este cheia secreta.
2. Calculeaza cheia publica $h = g^x \in G$.
3. Anunta cheia publica (G, g, h).

Protocol.

Bob:

1. Are de criptat un mesaj $m \in M$.
2. Alege un numar aleator y. Acesta este cheia lui efemera si va fi folosita doar in aceasta criptare.
3. Calculeaza $c_1 = g^y \in G$ si $c_2 = h^y m \in M$.
4. Transmite perechea (c_1, c_2) catre Alice.

Alice:

1. Afla $m = (c_1^x)^{-1} c_2$.

Teorema: *Protocolul Elgamal descifreaza mesajele in mod corect.*

Demonstratie:

$$(c_1^x)^{-1} c_2 = (g^{xy})^{-1}(g^{xy} m) = (g^{-xy} g^{xy})m = 1m = m.$$

Sunt mai multe realizari posibile pentru Elgamal.

Exemplu: Fie p un numar prim mare, de aproximativ 1024 biti. Presupunem ca $p - 1$ e divizibil printr-un numar prim mediu q, de aproximativ 160 biti. Se alege un element $g \in \mathbb{F}_p^\times$ de ordinul q, in modul urmator. Se alege un $r \in \mathbb{F}_p^\times$ astfel incat:

$$g = r^{\frac{p-1}{q}} \neq 1.$$

Atunci $|\langle g \rangle| = q$. Se ia $M = \mathbb{F}_p$ si $G = \langle g \rangle$.

Exemplu: Se poate lua $M = G = \mathbb{F}_{p^k}^\times$ pentru o reprezentare a corpului pentru care se cunoaste generatorul grupului $\mathbb{F}_{p^k}^\times$.

Exemplu: Se poate lua $M = \mathbb{Z}_{dp^k}$ cu $d \in \{1, 2\}$ si p prim impar, $G = \mathbb{Z}_{dp^k}^\times$, si g un generator al lui G.

Exemplu: Se poate lua G subgrup ciclic al unei curbe eliptice. Vom reveni asupra acestui exemplu.

Inainte de a termina, iata un protocol de schimb de cheie inrudit cu Elgamal. Este atat de apropiat, incat il prezentam tot in aceasta sectiune.

Protocolul Deffie-Hellman. Doua parti distante, Alice si Bob, dispun de un sistem de comunicare simetrica (Enc, Dec). Pentru a incepe o comunicare, ei trebuie sa stabileasca de comun acord o cheie k. In acest scop ei aleg impreuna un grup ciclic G si un generator g, cu $\langle g \rangle = G$. Acum incepe protocolul, care este absolut simetric:

1. Alice alege un numar random a, iar Bob alege un numar random b.
2. Alige trimite g^a catre Bob. In acelasi timp Bob trimite g^b catre Alice.
3. Alice calculeaza $k = (g^a)^b = g^{ab}$. Bob obtine acelasi $k = (g^b)^a = g^{ab}$.

Observatie: Securitatea depinde nu atat de structura grupului, cat de reprezentarea lui. Reprezentarea cea mai simpla $(\mathbb{Z}/n\mathbb{Z}, +, 0)$ nu garanteaza practic niciun pic de securitate. Fie $n = 100$. Bob si Alice aleg generatorul $g = 47$. Presupunem ca ei fac alegerile $a = 30$ si $b = 40$. In prima etapa Alice trimite 10 lui Bob iar Bob trimite 80 lui Alice. In a doua etapa Alice calculeaza $30 \cdot 80 = 0$ iar Bob calculeaza $40 \times 10 = 0$. Toate aceste calcule s-au facut mod100.

Cum se poate sparge acest protocol? Agenta Eva calculeaza intai $g^{-1} \bmod n = 47^{-1} \bmod 100$.

$$\begin{aligned} 100 &= 2 \cdot \underline{47} + \underline{6}, \\ \underline{47} &= 7 \cdot \underline{6} + \underline{5}, \\ \underline{6} &= \underline{5} + 1 \end{aligned}$$

$$1 = \underline{6} - \underline{5} = \underline{6} - (\underline{47} - 7 \cdot 6) = 8 \cdot \underline{6} - \underline{47} = -8 \cdot 2 \cdot \underline{47} - \underline{47} = (-17) \cdot 47 = 83 \cdot 47.$$

Deci $47^{-1} \bmod 100 = 83$. Acum Eva afla $a = 83 \cdot 10 = 30$ si calculeaza $k = 30 \cdot 80 = 0$.

In a doua incercare, Bob si Alice aleg grupul $(\mathbb{Z}_{101}^{\times}, \cdot, 1)$, grupul multiplicativ al unui corp finit. Acest grup este izomorf cu $(\mathbb{Z}_{100}, +, 0)$. Aratam ca $g = 2$ este generator multiplicativ. Cum $100 = 4 \cdot 25$, observam ca:

$$2^{50} \bmod 101 = 100 \neq 1,$$

$$2^{20} \bmod 101 = 95 \neq 1.$$

Ordinul lui 2 modulo 101 este 100, adica 2 este generator al grupului multiplicativ modulo 101.

De data aceasta Alice ii trimite $2^a = 2^{30} = 17 \bmod 101$ lui Bob iar Bob ii trimite $2^b = 2^{40} = 36 \bmod 101$ lui Alice. Dupa aceasta, Alice calculeaza $36^{30} = 1 \bmod 101$ iar Bob calculeaza $17^40 = 1 \bmod 101$. De data aceasta urmarirea conversatiei nu o ajuta pe pe Eva in niciun fel, deoarece pentru ea calculul logaritmului discret este o problema grea.

Urmatoarele paragrafe sunt dedicate unor criptari cu cheie publica mai exotice.

20 Rabin

Criptosistemul Rabin se bazeaza tot pe dificultatea factorizarii, mai exact pe dificultatea extragerii radacinii patrate modulo pq. Este mult mai rapid decat alte sisteme, dar introduce un grad de nedeterminare care l-a facut impropriu pentru criptarea automata a streamurilor. Aceasta ar fi fost aplicatia ideala pentru un criptosistem rapid.

Setup.

Alice:

1. Alege la intamplare doua numere prime $p = 4k + 3 \neq q = 4m + 3$.

2. Cheia secreta este perechea (p, q).

3. Cheia publica este (N, B) unde $N = pq$ iar $B \in \{0, 1, \ldots, N-1\}$ este ales la intamplare.

Protocol.

Bob:

1. Are de criptat un mesaj $m \in M$.

2. Calculeaza $c = m(m + B) \bmod N$.

Alice:

1. Afla $m = 2^{-1}(\sqrt{B^2 + 4c} - B) \bmod N$.

Teorema: *Alice descifreaza patru variante de mesaj, dintre care unul este mesajul clar.*

Demonstratie: Din

$$m^2 + mB - c = 0 \bmod N,$$

rezulta intr-adevar:

$$m = 2^{-1}(\sqrt{B^2 + 4c} - B) \bmod N.$$

Problemele vin de la extragerea radacinii patrate. Daca $\Delta = B^2 + 4c$, intai se scoate o radacina patrata din $\Delta \bmod p$ si una din $\Delta \bmod q$. Pentru fiecare radacina $a \bmod r$, elementul $r - a$ este si el o radacina. Explicit, radacina patrata din Δ se extrage in modul urmator:

- Se calculeaza radacinile mod$p$ si mod$q$. Pentru numere prime de forma $4k + 3$,

$$m_p = \Delta^{\frac{1}{4}(p+1)} \bmod p,$$

$$m_q = \Delta^{\frac{1}{4}(q+1)} \bmod q.$$

- Teorema Chineza a Resturilor poate fi exprimata si in modul urmator. Folosind Algoritmul lui Euclid extins, gasim $y_p, y_q \in \mathbb{Z}$ astfel incat:

$$y_p p + y_q q = 1.$$

atunci solutiile modulo N sunt:

$$\begin{aligned} r_1 &= (y_p \cdot p \cdot m_q + y_q \cdot q \cdot m_p) \bmod N, \\ r_2 &= N - r_1, \\ r_3 &= (y_p \cdot p \cdot m_q - y_q \cdot q \cdot m_p) \bmod N, \\ r_4 &= N - r_3. \end{aligned}$$

□

Exemplu: Fie $p = 127$ si $q = 131$, prin urmare $N = 16637$. Fie $B = 12345$. Daca $m = 4410$, atunci:

$$c = m(m + B) \bmod N = 4663.$$

Pentru decriptare, putem lucra si cu discriminantul redus.

$$\Delta' = \left(\frac{B^2}{4} + c\right) \bmod N = 1500.$$

$$\begin{aligned} \sqrt{\Delta'} &= \pm 22 \bmod 127, \\ \sqrt{\Delta'} &= \pm 37 \bmod 131. \end{aligned}$$

Cu Teorema Chineza obtinem valorile ± 3705 si ± 14373. Daca scadem $B/2 \bmod N$, obtinem: 4410, 5851, 15078, 16519. Primul mesaj este m. □

Din pacate nicio metoda de dezambiguare propusa nu a fost suficient de buna, astfel incat algoritmul nu s-a aplicat in practica pana acum.

21 Paillier

Incepem cu o varianta speciala a acestui sistem public de criptare.

Setup.

Alice:

1. Alege la intamplare doua numere prime $p \neq q$ astfel incat $\gcd(pq, (p-1)(q-1)) = 1$. Asta se intampla automat daca p si q au lungimi binare egale.
2. Cheia publica este N unde $N = pq$.
3. Cheia secreta este un d astfel incat:

$$\begin{aligned} d &= 1 \bmod N, \\ d &= 0 \bmod (p-1)(q-1). \end{aligned}$$

   Acest d se obtine prin Teorema Chineza a Resturilor.

Protocol.

Bob:

1. Are de criptat un mesaj m cu $0 \leq m < N$.
2. Alege $r \in \mathbb{Z}_{N^2}^{\times}$.
3. Calculeaza si trimite $c = (1+N)^m r^N \bmod N^2$.

Alice:

1. Calculeaza $t = c^d \bmod N^2$.
2. Afla $m = (t-1)/N$ prin impartire cu rest in $\mathbb{Z}$.

Teorema: *Algoritmul de mai sus duce la decriptarea corecta a mesajului clar.*

Demonstratie:

$$t = c^d \bmod N^2 = (1+N)^{md} r^{dN} \bmod N^2.$$

Dar cum $(p-1)(q-1) | d$, $r^{dN} = 1$ deoarece r este element intr-un grup cu

$$\varphi(N^2) = \varphi(p^2q^2) = p^2q^2(1-\frac{1}{p})(1-\frac{1}{q}) = pq(p-1)(q-1) = N(p-1)(q-1)$$

elemente. Deci:

$$t = (1+N)^{md} \bmod N^2 = 1 + mdN \bmod N^2.$$

Ultima egalitate are loc deoarece ceilalti termeni din binomul lui Newton sunt divizibili cu N^2. Avand in vedere ca $d = 1 \bmod N$,

$$t = 1 + mN \bmod N^2.$$

Asadar,

$$m = \frac{t-1}{N}.$$

□

Iata si o forma mai generala a algoritmului. Avem nevoie de urmatoarea definitie:

Definitie: Functia L se defineste ca:

$$L(x) = \frac{x-1}{N}.$$

Aici impartirea este impartire cu rest in $\mathbb{Z}$. Functia L joaca rolul unui logaritm discret in $\mathbb{Z}_{N^2}^{\times}$.

Setup.

Alice:

1. Alege la intamplare doua numere prime $p \neq q$ astfel incat $\gcd(pq, (p-1)(q-1)) = 1$. Fie $N = pq$ si $\lambda = \text{lcm}(p-1, q-1)$.
2. Alege la intamplare $g \in \mathbb{Z}_{N^2}^{\times}$ astfel incat $N | \text{ord}(g)$.
3. Fie $\mu = (L(g^{\lambda} \bmod N^2))^{-1} \bmod N$.
4. Cheia publica este perechea (N, g).
5. Cheia secreta este perechea (λ, μ).

Protocol.

Bob:

1. Are de criptat un mesaj m cu $0 \leq m < N$.
2. Alege $r \in \mathbb{Z}_{N^2}^{\times}$ random.
3. Calculeaza si trimite $c = g^m r^N \bmod N^2$.

Alice:

1. Calculeaza $m = L(c^{\lambda} \bmod N^2)\mu \bmod N$.

Teorema: *Algoritmul de mai sus duce la decriptarea corecta a mesajului clar.*

Demonstratie:

$$L(c^{\lambda} \bmod N^2)\mu \bmod N = L(g^{\lambda m} r^{\lambda N} \bmod N^2)\mu \bmod N =$$

$$= L(g^{\lambda m} \bmod N^2)(L(g^{\lambda} \bmod N^2))^{-1} \bmod N = m\mu^{-1}\mu = m.$$

□

Se observa ca daca p si q sunt alese de aceeasi lungime (in practica intre 1024 si 2048 biti), $g = N + 1$, $\lambda = \varphi(N)$ si $\mu = \varphi(N)^{-1} \bmod N$, se obtine prima varianta a algortmului.

De ce functia $L(x)$ joaca rolul unui logaritm discret? Explicatia este urmatoarea. Din:

$$(1+N)^x = 1 + Nx \bmod N^2,$$

rezulta pentru $y = (1+N)^x \bmod N^2$ ca:

$$x = \frac{y-1}{N} \bmod N,$$

$$L((1+N)^x \bmod N^2) = x,$$

pentru orice $x \in \mathbb{Z}_N$. Asadar securitatea criptosistemului Paillier este asigurata in acelasi timp de dificultatea factorizarii si de o anumita forma a logaritmului discret.

Observam ca sistemul Paillier are anumite proprietati homomorfice, cum ar fi:

$$D(E(m_1 r_1)E(m_2 r_2)) \bmod N^2) = (m_1 + m_2) \bmod N,$$

$$D(E(m_1 r_1)^k \bmod N^2) = k m_1 \bmod N.$$

Aceste proprietati il fac potrivit pentru aplicatii care folosesc proprietatile homomorfice, cum ar fi votul electronic sau calculul secret partajat.

22 Goldwasser-Micali

Este o metoda de criptare a bitilor bazata pe dificultatea lui QUADRES, respectiv a factorizarii.

Setup.

Alice:

1. Alege la intamplare doua numere prime $p \neq q$. Fie $N = pq$.
2. Alege $k \in \mathbb{Z}_N$ astfel incat:
$$\left(\frac{k}{p}\right) = -1 \ \wedge \ \left(\frac{k}{q}\right) = -1.$$
Asta se poate face alegand non-resturi modulo p si q si aplicand Teorema Chineza. O idee mai rapida este sa alegem $p = 3 \bmod 4$, $q = 3 \bmod 4$ si $k = N - 1$. Numarul k se numeste pseudopatrat. In $\mathbb{Z}_N$ are simbolul Legendre al unui patrat, dar nu este un patrat.
3. Cheia publica este perechea (N, k).
4. Cheia secreta este perechea (p, q).

Protocol.

Bob:

1. Are de criptat un mesaj $m \in \{0, 1\}$.
2. Alege $r \in \mathbb{Z}_N^\times$ random.
3. Calculeaza si trimite $c = mr^2 \bmod N$.

Alice:

1. Calculeaza $\left(\frac{c}{p}\right) = c^{\frac{p-1}{2}} \bmod N$. Daca c este un patrat, $m = 1$; altfel, $m = 0$.

Este evident faptul ca aceasta procedura este nedeterminista, si ca se cripteaza respectiv se decripteaza corect fiecare bit.

Part IV
Metode de atac

23 Atacuri simple la RSA

Teorema: *Daca cunoastem N, e, d, atunci putem factoriza N, si vom putea calcula cheia secreta pentru oricare alta cheie publica.*

Demonstratie: Stim ca $ed-1 = s(p-1)(q-1)$. Alegem un $x \in \mathbb{Z}_N^\times$, stim ca:

$$x^{ed-1} = 1 \bmod N.$$

De fapt $ed-1$ este multiplu al lui $\lambda(N)$, si $ed-1$ este par. Scriem $ed-1 = 2^t h$, unde h impar.

Fie $b = x^h \bmod N$. Acest element are ca ordin 1 sau o putere a lui 2. Stim de asemeni ca ordinul este cel mai mic multiplu comun ale ordinelor in $\mathbb{Z}_p^\times$ si respectiv in $\mathbb{Z}_q^\times$. Speram ca aceste ordine sunt diferite, lucru care are o probabilitate destul de mare.

Calculam $g = \gcd(b-1, N)$. Daca $g \neq 1$, am obtinut un multiplu de p sau de q. Daca $g = 1$, inlocuim pe b cu $b^2 \bmod N$, si calculam din nou $\gcd(b-1, N)$.

Daca la un moment dat $g = N$, atunci ordinele lui $x^h \bmod p$ si $x^h \bmod q$ erau egale, si trebuie sa alegem alt x.

□

Exemplu: Fie $N = 21$, cunoastem $e = d = 5$. Atunci $ed - 1 = 24 = 3 \cdot 8$. Alegem $x = 2$. Cum $\gcd(2, 21) = 1$, aceasta alegere nu ne ajuta direct sa gasim un divizor. Acum $b = 2^3 = 8$, $b-1 = 7$, $\gcd(7, 21) = 7$ si $21 = 7 \cdot 3$. □

Teorema: *Daca stim N si $\varphi(N)$, putem factoriza N.*

Demonstratie:

$$\varphi(N) = (p-1)(q-1) = pq - p - q + 1 = N - (p+q) + 1,$$

$$p + q = N + 1 - \varphi(N).$$

Asadar, p si q sunt solutia ecuatiei de gradul 2:

$$X^2 - (N + 1 - \varphi(N))X + N = 0.$$

□

Teorema: *Daca un mesaj m este trimis folosind acelasi modul N si chei publice e_1 si e_2 astfel incat $\gcd(e_1, e_2) = 1$, atunci se poate descifra mesajul fara a cunoaste nicio cheie secreta.*

Demonstratie: Fie asadar $\gcd(e_1, e_2) = 1$ si mesajele criptate:

$$\begin{aligned} c_1 &= m_1^{e_1} \bmod N, \\ c_2 &= m_2^{e_2} \bmod N. \end{aligned}$$

Atacatorul calculeaza:

$$\begin{aligned} t_1 &= e_1^{-1} \bmod e_2, \\ t_2 &= \frac{e_1 t_1 - 1}{e_2}. \end{aligned}$$

Deci $e_2 t_2 = e_1 t_1 - 1$. Asadar:

$$c_1^{t_1} c_2^{-t_2} = m^{e_1 t_1} m^{-e_2 t_2} = m^{1 + e_2 t_2 - e_2 t_2} = m \mod N.$$

□

Teorema: *Fie m un mesaj cu $0 \leq m < \min(N_1, \ldots, N_e)$. Folosind aceeasi cheie e dar un numar de e moduli relativ primi $N_1, \ldots, N_e$, se calculeaza mesajele cifrate:*

$$c_j = m^e \bmod N_j.$$

Din $c_1, \ldots, c_e$ se poate reconstitui m.

Demonstratie: Din Teorema Chineza a Resturilor si faptul ca modulii N_j sunt primi intre ei, gasim un c astfel incat:

$$c = c_j \bmod N_j,$$

pentru toti $j = 1, \ldots, e$ si $0 \leq c < N_1 N_2 \ldots N_e$. Cum $0 \leq m^e < N_1 N_2 \ldots N_e$ si $c = m^e \bmod N_1 N_2 \ldots N_e$ rezulta ca $c = m^e$, deci $m = \sqrt[e]{c}$. □

Observatie: Acest atac este cea mai simpla forma a atacului Hastad. Intr-o sectiune ulterioara vom prezenta forme mai complicate in cadrul metodei Coppersmith.

Observatie: Acest atac functioneaza si cu receptionarea a numai doua mesaje cu aceeasi cheie si moduli diferiti, in cazul in care valoarea mesajului nu este prea mare. Trebuie ca $m^e < N_1 N_2$ adica $m < \sqrt[e]{N_1 N_2}$. □

Morala: Aceste exemple arata ca primitiva RSA are anumite slabiciuni de securitate native. De aceea este necesara imunatatirea lui cu anumite masuri de precautie, care sa il faca mai putin determinist. Dintre aceste masuri amintim:

1. Schimbarea random a modulului N si a cheii publice de criptare e, daca se poate la fiecare noua transmisie.
2. Folosirea unor chei relativ mari, gen $e = 65537$, si a unor numere prime p si q foarte mari.
3. Completarea mesajului m cu un bloc de biti random egal ca lungime inaintea criptarii. Asa repetarea mesajului nu mai produce acelasi mesaj cifrat, ci mereu altul.

24 Atacul Wiener

Atacul Wiener este o metoda de atac asupra RSA. Metoda se poate aplica in cazul exponentilor de criptare d relativ mici. Ce inseamna *relativ mic* va rezulta din teorie.

Descompunerea in fractii continue: Fie $\alpha \in \mathbb{R}$. Construim sirurile $(\alpha_n) \subset \mathbb{R}$ si $(a_n), (p_n), (q_n) \subset \mathbb{Z}$ recursiv, in modul urmator:

$$\alpha_0 = \alpha,\ p_0 = a_0,\ q_0 = 1,$$

$$p_1 = a_0 a_1 + 1,\ q_1 = a_1,$$

$$a_i = [\alpha_i],$$

$$\alpha_{i+1} = \frac{1}{\alpha_i - a_i},$$

$$p_i = a_i p_{i-1} + p_{i-2},\ i \geq 2,$$

$$q_i = a_i q_{i-1} + q_{i-2},\ i \geq 2.$$

Intregii (a_i) formeaza fractia continua a lui α. Primul poate fi negativ, ceilalti sunt numere naturale. Acest sir este periodic daca si numai daca α este numar rational sau solutie reala a unei ecuatii de gradul 2 cu coeficienti intregi. Fractiile

$$\frac{p_i}{q_i}$$

se numesc convergentii lui α. Sirul $p_i/q_i \to \alpha$ si pentru orice i, $\gcd(p_i, q_i) = 1$.

Un rol crucial este jucat de urmatorul:

Fapt: Daca $p, q \in \mathbb{Z}$, si are loc:

$$\left|\alpha - \frac{p}{q}\right| \leq \frac{1}{2q^2},$$

atunci exista un i astfel incat $p = p_i$ si $q = q_i$, adica p/q este un convergent al lui α.

Facem urmatoarele presupuneri: $N = pq$, numerele prime p si q sunt apropiate ca marime, adica $q < p < 2q$, iar d este relativ mic, adica riguros exprimat:

$$d < \frac{1}{3}\sqrt[4]{N}.$$

De asemeni cheia publica e este mai mica decat $\Phi = (p-1)(q-1)$. Stim ca exista un k astfel incat $ed - k\Phi = 1$, adica:

$$\Big|\frac{e}{\Phi} - \frac{k}{d}\Big| = \frac{1}{d\Phi}.$$

Numerele Φ si N pot fi considerate apropiate, deoarece:

$$|N - \Phi| = |p + q - 1| < 3\sqrt{N}.$$

Observam ca $\frac{e}{N}$ este o aproximatie stransa a lui $\frac{k}{N}$:

$$\Big|\frac{e}{N} - \frac{k}{d}\Big| = \Big|\frac{ed - kN}{dN}\Big| = \Big|\frac{ed - k\Phi - Nk + k\Phi}{dN}\Big| = \Big|\frac{1 - k(N - \phi)}{dN}\Big| \leq$$

$$\leq \Big|\frac{3k\sqrt{N}}{dN}\Big| = \frac{3k}{d\sqrt{N}} < \frac{1}{3d^2} < \frac{1}{2d^2}.$$

Aceasta, deoarece din $e < \Phi$ rezulta:

$$k < d < \frac{1}{3}\sqrt[4]{N}, \quad \frac{k}{d} < \frac{\sqrt[4]{N}}{3d}.$$

Aplicand ultima inegalitate obtinem:

$$\frac{3k}{d\sqrt{N}} < \frac{3}{3d}\frac{\sqrt[4]{N}}{\sqrt{N}} = \frac{1}{d\sqrt[4]{N}} < \frac{1}{3d^2}.$$

Dar $\gcd(k, d) = 1$, deci k/d este fractie simplificata.

Concluzie: *$\frac{k}{d}$ este unul din convergentii lui $\frac{e}{N}$.*

Exemplu:

$N = 9\ 449\ 868\ 410\ 449$,

$e = 6\ 792\ 605\ 526\ 025$.

Nu stim daca metoda va da rezultate, dar stim ca trebuie sa cautam printre acei d care sunt:

$$d < \frac{1}{3}\sqrt[4]{N} < 584.$$

Calculam convergentii lui $\alpha = e/N$. Obtinem:

$$1, \frac{2}{3}, \frac{3}{4}, \frac{5}{7}, \frac{18}{25}, \frac{23}{32}, \frac{409}{569}, \frac{1659}{2308}.$$

Gasim ca $d = 569$ verificand ca:

$$(m^e)^d = m \bmod N,$$

pentru cateva valori m alese la intamplare.

25 Metoda Coppersmith

Atacurile cu metoda Coppersmith se adreseaza sistemului RSA.

Teorema 1 a lui Coppersmith: *Fie N be un intreg si $f \in \mathbb{Z}[x]$ un polinom monic de grad d. Fie $X = N^{\frac{1}{d}-\varepsilon}$ pentru $\frac{1}{d} > \varepsilon > 0$. Data fiind perechea $\langle N, f\rangle$ atacatorul Eve poate gasi efectiv toti intregii $x_0 < X$ care satisfac $f(x_0) \equiv 0 \pmod N$. Timpul este dominat de timpul necesar algoritmului LLL pe o latice de dimensiune $O(w)$ cu $w = \min\left\{\frac{1}{\varepsilon}, \log_2 N\right\}$.*

Atacul Hastad. La atacul Hastad s-a propus ca masura de precautie padding cu random biti. Acum vom vedea ca in anumite situatii nici aceasta masura nu este suficienta. Presupunem ca atacatorul receptioneaza coduri $c_i = f_i(m)^e$ cu $1 \leq i \leq k$, unde functiile f_i sunt liniare. Cu alte cuvinte, mesajul m a fost completat cu o adresa sau cu un numar de ordine inainte de a fi trimis - de exemplu $m_i = i \cdot 2^m + m$ este cifrat cu cheia e, apoi trimis adresantului i. Daca numarul de adresanti este suficient de mare, mesajul poate fi descifrat.

Teorema lui Hastad: *Fie $N_1, \ldots, N_k$ intregi relativ primi si $N_{\min} = \min(N_i)$. Fie $g_i \in \mathbb{Z}_{N_i}[x]$ un numar de k polinoame de grad maxim q. Presupunem ca exista un unic $m < N_{\min}$ care satisface $g_i(m) = 0 \bmod N_i$ pentru toti $1 \leq i \leq k$. Mai mult, presupunem $k > q$. Atunci exista un algoritm eficient care calculeaza m.*

Demonstratie: Fiindca numerele N_i sunt relativ prime, se poate aplica Teorema Chineza a Resturilor pentru a calcula coeficienti T_i astfel incat $T_i = 1 \bmod N_i$ si $T_i = 0 \bmod N_j$ pentru $j \neq i$. Fie $g(x) = \sum T_i g_i(x)$. Stim ca $g(m) = 0 \bmod \prod N_i$. Gradul lui g este cel mult q. Cu teorema lui Coppersmith putem calcula toate radacinile intregi x_0 care satisfac $g(x_0) = 0 \bmod \prod N_i$ cu:

$$|x_0| < \Big(\prod N_i\Big)^{\frac{1}{q}}.$$

Dar stim ca:

$$m < N_{\min} < \Big(\prod N_i\Big)^{\frac{1}{k}} < \Big(\prod N_i\Big)^{\frac{1}{q}}.$$

□

Atacul Franklin-Reiter. Fie (N, e) cheia publica a lui Alice. Fie $m_1, m_2 \in \mathbb{Z}_N$doua mesaje care satisfac $m_1 = f(m_2) \bmod N$, unde $f(x) = ax + b\mathbb{Z}_N[x]$ este un polinom liniar cunoscut. De exemplu, Bob primeste mesajul inapoi impreuna cu o mentiune de eroare, si retrimite intregul mesaj la aceeasi adresa. In total, Bob trimite doua mesaje criptate c_1 si c_2.

Teorema Franklin-Reiter: *Daca (N, e) este o cheie publica RSA si $m_1 \neq m_2 \in \mathbb{Z}_N$ sunt doua mesaje astfel incat $m_1 = f(m_2) \bmod N$ pentru un polinom liniar $f = ax + b \in \mathbb{Z}_N[x]$ cu $b \neq 0$, atunci m_1 si m_2 se pot recupera in timp patratic in $\log_2 N$ si e.*

Demonstratie: Cum $c_1 = m_1^e \bmod N$, stim ca m_2 este radacina comuna a polinoamelor $g_1(x) = f(x)^e - c_1 \in \mathbb{Z}_N[x]$ si $g_2(x) = x^e - c_2 \in \mathbb{Z}_N[x]$. Factorul liniar $x - m_2$ divide ambele polinoame. Deci atacatorul calculeaza $\gcd(g_1(x), g_2(x))$ folosind algoritmul lui Euclid. In cazul in care acest polinom este liniar, l-am gasit pe m_2, iar din exuatia liniara il gasim si pe m_1. Daca nu este liniar, trebuie descompus. □

Atacul lui Coppersmith. Acest atac se poate aplica atunci cand operatia de padding se face cu prea putini biti random. Eve intercepteaza mesajul si il opreste. Bob vede ca Alice nu raspunde si retrimite mesajul criptat, de data aceasta cu un alt padding.

Teorema 2 a lui Coppersmith: *Fie (N, e) o cheie publica RSA, unde N are o lungime de n biti. Fie $m = [n/e^2]$. Fie $M \in \mathbb{Z}_N$ un mesaj de lungime cel mult $n - m$ biti. Fie $m_1 = 2^m M + r_1$ si $m_2 = 2^m M + r_2$ unde $r_1 \neq r_2$ si $0 \leq r_1, r_2 \leq 2^m$. Daca un atacator are criptarile c_1 si c_2, dar nu are r_1 si r_2, poate recupera mesajul M in mod eficient.*

Demonstratie: Fie $g_1(x, y) = x^e - c_1$ si $g_2(x, y) = (x + y)^e - c_2$. Stim ca pentru $y = r_2 - r_1$ aceste polinoame au $x = m_1$ ca radacina comuna. Cu alte cuvinte, $\Delta = r_2 - r_1$ este radacina a

rezultantului $\text{res}_x(g_1, g_2) \in \mathbb{Z}_N[y]$. Mai mult,

$$|\Delta| < 2^m < N^{\frac{1}{e^2}}.$$

Asadar Δ poate fi gasit cu metoda Coppersmith. Odata Δ cunoscut, se poate afla m_2 cu metoda Franklin-Reiter, iar din m_2 se afla M. □

Explicatie: Rezultantul este determinantul Matricii Sylvester. Daca $A(x, y)$ are grad $d = 3$ in x si $B(x, y)$ are grad $e = 2$ in x, atunci rezultantul $\text{res}_x(A, B)$ este determinantul matricii:

$$\begin{pmatrix} a_0 & 0 & b_0 & 0 & 0 \\ a_1 & a_0 & b_1 & b_0 & 0 \\ a_2 & a_1 & b_2 & b_1 & b_0 \\ a_3 & a_2 & 0 & b_2 & b_1 \\ 0 & a_3 & 0 & 0 & b_2 \end{pmatrix}$$

unde a_i si b_j sunt polinoame in y. Sunt e coloane de a_i si d coloane de b_j. Daca P_k este spatiul vectorial al polinoamelor de grad strict mai mic decat k, atunci Matricea Sylvester este matricea aplicatiei liniare:

$$\varphi : P_e \times P_d \to P_{d+e},$$
$$\varphi(P, Q) = AP + BQ,$$

scrisa relativ la baza data de puterile lui x in ordine descrescatoare, deci astfel incat $A = a_0x^d + \cdots + a_d$ si $B = b_0x^e + \cdots + b_e$. □

26 Algoritmi de factorizare

Multi algoritmi de factorizare depind de principiul lui Dirichlet, adica de proiectia unor observabile pe niste criterii de clasificare - de obicei niste clase de resturi - in scopul realizarii unei coliziuni. Cu alte cuvinte, se cauta observabile care cad in aceeasi clasa. Calculul de probabilitate corespunzator poarta numele generic de *Paradox al datei de nastere*. Acest nume se bazeaza pe urmatorul experiment: daca intr-o clasa de elevi, fiecare isi declara data nasterii, probabilitatea unei coliziuni este surprinzator de mare. Pentru 24 de elevi, probabilitatea depaseste deja 1/2.

Rezultatul urmator este important si pentru gasirea coliziunilor la functii hash:

Paradoxul datei de nastere. *Daca n variabile aleatoare independente pot lua m valori posibile, probabilitatea de nu se inregistra nicio coliziune este:*

$$\Big(1 - \frac{1}{m}\Big)\Big(1 - \frac{2}{m}\Big) \dots \Big(1 - \frac{n-1}{m}\Big) = \prod_{i=1}^{n-1} \Big(1 - \frac{i}{m}\Big) \simeq$$

$$\simeq \prod_{i=1}^{n-1} e^{\frac{-i}{m}} = e^{-\sum\limits_{i=1}^{n-1} \frac{i}{m}} = e^{-\frac{n(n-1)}{2m}} \simeq e^{-\frac{1}{2m}(n-\frac{1}{2})^2}.$$

Observam ca daca $n > 1.2\sqrt{m}$ atunci probabilitatea unei coliziuni este $> 1/2$.

Algoritmul naiv. Se verifica daca numerele d cu $2 \le d \le \sqrt{n}$ verifica $d|n$. Daca a gasit un divizor se opreste. Algoritmul se poate relua pentru numerele d si n/d. In cazul modulilor RSA, care au forma $N = pq$, daca am gasit p, am rezolvat problema factorizarii. Fie m lungimea lui n in biti. Atunci algoritmul clasic dureaza $O(\sqrt{n}) = O(2^{\frac{m}{2}})$.

Algoritmul Pollard's Rho. Foloseste paradoxul datei de nastere in urmatorul mod: Fie p prim necunoscut, care divide n. Consideram elemente $x_1, \dots, x_k \in \mathbb{Z}_n$ cu $k > 1.2\sqrt{p}$. Atunci, cu probabilitate $> 1/2$, exista $i \neq j$ cu $x_i = x_j \bmod p$. Daca $x_i \neq x_j$, $d = \gcd(n, x_i - x_j)$ va fi un divizor al lui n.

Daca folosim un sir pseudo-aleator $x_{i+1} = f(x_i) \bmod p$ cu f polinom, acesta are si proprietarea suplimentara ca daca $a = b \bmod p$ atunci $f(a) = f(b) \bmod p$. Aparitia coliziunii poate fi exprimata intr-o diagrama de tipul urmator:

$$\begin{array}{ccc} x_5 & \rightarrow & x_6 \\ \uparrow & & \downarrow \\ x_4 & & x_7 \\ \uparrow & & \downarrow \\ x_3 & \leftarrow & x_8 \\ \uparrow & & \\ x_2 & & \\ \uparrow & & \\ x_1 & & \\ \uparrow & & \\ x_0 & & \end{array}$$

Numele Pollard's Rho vine de la asemanarea acestei diagrame cu litera greceasca ρ. Problema acestei scheme este ca dupa calcularea unui x_i, trebuie facute toate $\gcd(n, x_i - x_j)$ cu $j < i$, in speranta ca gasim o situatie in care acesta este diferit atat de 1 cat si de n. (In cazul in care obtinem n, reluam procedeul cu un alt x_1.) Acesta este un volum foarte mare de calcul. De aceea se aplica urmatorul rezultat, pe care Donald Knuth l-a atribuit lui Robert W. Floyd:

Lema lui Floyd: *Daca pentru $i < j$ are loc $x_i = x_j \bmod p$, atunci exista un $k \in \{i, i+1, \ldots, j-1\}$ cu $x_k = x_{2k} \bmod p$.*

Demonstratie: Conform ipotezei exista un segment initial de lungime i si un ciclu de lungime $j-i$. Asadar fie se obtine o coliziune cand $k \geq i$ si diferenta $2k-k = k$ este un multiplu de $j-i$. Fie $k = i$. Daca $2k = j$, este ok. Altfel, este $2k - k = k \neq 0 \bmod (j - i)$. Incepem sa adunam unitati la k. Dupa cel mult $j - i - 1$ pasi, este $2k - k = 0 \bmod (j - i)$. Asadar $k \leq i + (j - i - 1) = j - 1$.

□

Iata acest algoritm reprezentat schematic:

input n;
choose $x < n$;
$y = x$;
repeat
    $x = f(x) \bmod n$;
    $y = f(y) \bmod n$;
    $y = f(y) \bmod n$;
    $g = \gcd(n, x - y)$;
until $((x \neq y)$ **and** $(g > 1))$;
return g;

In practica se foloseste functia $f(n) = n^2 + 1$ sau mai general $f(n) = n^2 + a$.

Teorema: *Pollard's Rho gaseste un divizor al lui n in timp $O(\sqrt[4]{n}) = O(2^{\frac{m}{4}})$.*

Demonstratie: Daca elementele aleatoare din program sunt respectate, paradoxul datei de nastere apare in $O(\sqrt{p})$ pasi. Dar $p \leq \sqrt{n}$. □

Pollard's $p-1$. Fie p un numar prim care divide n. Exista un B astfel incat $B! = (p-1)k$. Deci:

$$2^{B!} = 2^{(p-1)k} = 1 \bmod p,$$

in corpul $\mathbb{Z}_p$. Deci $\gcd(2^{B!} - 1, n)$ va fi factor al lui n, daca este mai mic decat n. Algoritmul are urmatoarea forma:

$i = 1$;

choose $a \in \{2, 3, \ldots, p-1\}$;

loop

$i = i + 1$;

$a = a^i \bmod n$;

$b = \gcd(a - 1, n)$;

if $(1 < b < n)$ **return** b;

Obsevatie: In worst case $n = pq$ iar numerele prime $x = p$ si $x = q$ sunt de marime apropiata si au proprietatea ca $x = 2y$, unde y este un numar prim. In aceasta situatie Pollard's $p - 1$ nu este mai bun decat algoritmul naiv de factorizare. In toate situatiile in care x are forma $2p'q'k$, deci este divizibil prin cel putin doua numere prime diferite de 2, Pollard's $p - 1$ este cel putin la fel de bun ca Pollard's Rho.

Fermat. Fie $n = pq$ unde $p < q$ sunt impare si diferite de 1. Atunci urmatoarele numere sunt intregi:

$$x = \frac{p+q}{2} \quad , \quad y = \frac{p-q}{2},$$

si, mai mult,

$$n = (x + y)(x - y) = x^2 - y^2.$$

Cum $y^2 \geq 0$ trebuie sa fie intotdeauna $x \geq \sqrt{n}$. Aceste observatii se aplica in urmatorul algoritm:

input n;

$x = [\sqrt{n}] + 1$; $z = x^2 - n$;

repeat

if $y = \sqrt{z} \in \mathbb{N}$ **return** $x - y$;

$x = x + 1$;

$z = z + 2x - 1$;

untill $x = n$;

Se observa ca acest algoritm este eficient in cazul numerelor RSA de forma $n = pq$ numai daca ambii factori primi sunt apropiati de $\sqrt{n}$.

Sita patratica. Ideea algoritmului este similara cu algoritmul lui Fermat. Anume, sa gasim x si y cu $x^2 - y^2 = kn$, adica:

$$x^2 = y^2 \bmod n.$$

Atunci, daca $x \neq \pm y \bmod n$, $\gcd(x \mp y, n)$ este un divizor propriu al lui n. Cum un patrat are modulo $n = pq$ cel putin 4 radacini patrate, o radacina buna se gaseste cu probabilitate de cel putin 1/2.

In acest scop cautam o baza $\{p_1, p_2, \ldots, p_k\}$, deci o multime de numere prime cunoscute relativ mici, astfel incat pentru niste numere a_i cu $i = 1, \ldots, k$, numerele $a_i^2 - n$ sa admita numai aceste numere prime in descompunerea lor. Exista euristici pentru gasirea acestor reprezentari. Asadar:

$$\begin{aligned}
a_1^2 - n &= p_1^{j_{1,1}} p_2^{j_{1,2}} \ldots p_k^{j_{1,k}} \\
a_2^2 - n &= p_1^{j_{2,1}} p_2^{j_{2,2}} \ldots p_k^{j_{2,k}} \\
\vdots & \qquad\qquad \vdots \\
a_k^2 - n &= p_1^{j_{k,1}} p_2^{j_{k,2}} \ldots p_k^{j_{k,k}}
\end{aligned}$$

Daca gasim o submultime $I \subseteq \{1, 2, \ldots, k\}$ astfel incat:

$$\prod_{i \in I}(a_i^2 - n) = \prod_{i \in I} p_1^{j_{i,1}} p_2^{j_{i,2}} \ldots p_k^{j_{i,k}} = p_1^{\sum_{i \in I} j_{i,1}} p_2^{\sum_{i \in I} j_{i,2}} \ldots p_k^{\sum_{i \in I} j_{i,k}}$$

sa fie un patrat, adica exponentii numerelor prime sa fie numere pare. In cazul in care am gasit o asemenea submultime I, definim:

$$x = \prod_{i \in I} a_i.$$

Atunci:

$$\prod_{i \in I}(a_i^2 - n) = y^2,$$

$$\prod_{i \in I} a_i^2 = y^2 \bmod n,$$

$$x^2 = y^2 \bmod n.$$

Pentru a gasi submultimea I, cautam un vector $(z_1, z_2, \ldots z_k) \in \{0, 1\}^k$ care sa rezolve urmatorul sistem de ecuatii liniare peste corpul $\mathbb{F}_2$:

$$\begin{aligned} b_{1,1}z_1 + \cdots + b_{k,1}z_k &= 0 \\ \vdots & \quad \vdots \\ b_{1,k}z_1 + \cdots + b_{k,k}z_k &= 0 \end{aligned}$$

unde fiecare $b_{u,v} = j_{u,v} \bmod 2$.

27 Algoritmi pentru logaritmul discret

In cele ce urmeaza, fie p un numar prim si g o radacina primitiva modulo p, adica un generator al grupului multiplicativ $\mathbb{F}_p^\times$. Se da un $y \in \mathbb{F}_p^\times$ si se cere x astfel incat $y = g^x$. Exista si notatia $x = \log_g y \bmod p$.

Baby Step - Giant Step. In acest algoritm se tine seama de faptul ca orice $x < p$ are o reprezentare:

$$x = \lceil \sqrt{p} \rceil x_1 + x_2, \quad \text{cu} \quad 0 \leq x_1, x_2 \leq \lfloor \sqrt{p} \rfloor.$$

Deci:

$$y = g^x = g^{\lceil \sqrt{p} \rceil x_1 + x_2} = \left(g^{\lceil \sqrt{p} \rceil}\right)^{x_1} \cdot g^{x_2},$$

$$y \cdot \left(g^{-1}\right)^{x_2} = \left(g^{\lceil \sqrt{p} \rceil}\right)^{x_1}.$$

Asadar calculam $z = g^{-1} \bmod p$ cu algoritmul inversului modular, si calculam $h = g^{\lceil \sqrt{p} \rceil} \bmod p$ folosind algoritmul de exponentiere modulara rapida. In acest moment se construiesc doua liste:

$$L_1 = \{(x_1, h^{x_1}) \,|\, x_1 = 0, 1, \ldots, \lfloor \sqrt{p} \rfloor\},$$

$$L_2 = \{(x_2, yz^{x_2}) \,|\, x_2 = 0, 1, \ldots, \lfloor \sqrt{p} \rfloor\}.$$

Se cauta o coincidenta de forma $(x_1, w) \in L_1$, $(x_2, w) \in L_2$. In acest caz $x = \lceil \sqrt{p} \rceil x_1 + x_2$ este logaritmul discret cautat.

Teorema: *Algoritmul Baby Step - Giant Step are un timp de lucru de* $O(\sqrt{p}) = O(2^{\frac{m}{2}})$ *unde* m *este lungimea lui* p *in biti.*

Pollard's Rho pentru Logaritmul Discret. Definim urmatoarele intervale discrete:

$$\begin{aligned} M_1 &= \left\{1, \ldots, \left\lfloor \frac{p}{3} \right\rfloor\right\}, \\ M_2 &= \left\{\left\lfloor \frac{p}{3} \right\rfloor + 1, \ldots, \left\lfloor \frac{2p}{3} \right\rfloor\right\}, \\ M_3 &= \left\{\left\lfloor \frac{2p}{3} \right\rfloor + 1, \ldots, p-1\right\}. \end{aligned}$$

Se observa ca aceasta este o partitie $M_1 \cup M_2 \cup M_3 = \{1, \ldots, p-1\}$ in segmente aproximativ egale. Alegem la intamplare un numar a si consideram elementul $z_0 = g^a y^0 \bmod p$. Se genereaza un sir pseudo-random z_i in modul urmator:

$$z_{i+1} = \begin{cases} z_i^2 \bmod p, & z_i \in M_1, \\ z_i \cdot g \bmod p, & z_i \in M_2, \\ z_i \cdot y \bmod p, & z_i \in M_3. \end{cases}$$

Asta inseamna ca $z_i = g^{a_i} y^{b_i}$ unde $a_0 = a$, $b_0 = 0$ si in general:

$$a_{i+1} = \begin{cases} 2a_i \bmod (p-1), & z_i \in M_1, \\ a_i + 1 \bmod (p-1), & z_i \in M_2, \\ a_i \bmod (p-1), & z_i \in M_3, \end{cases} \quad b_{i+1} = \begin{cases} 2b_i \bmod (p-1), & z_i \in M_1, \\ b_i \bmod (p-1), & z_i \in M_2, \\ b_i + 1 \bmod (p-1), & z_i \in M_3. \end{cases}$$

Folosind Lema lui Floyd cautam o pereche $i = k$ si $j = 2k$ astfel incat $z_i = z_j \bmod p$. Asta se intampla cu mare probabilitate pentru un $k < 1.2\sqrt{p}$. Asadar:

$$g^{a_i + xb_i} = g^{a_j + xb_j} \bmod p,$$

adica:

$$\begin{aligned} a_i + xb_i &= a_j + xb_j \bmod (p-1), \\ x(b_i - b_j) &= a_j - a_i \bmod (p-1). \end{aligned}$$

Cum $p-1$ nu este un numar prim, se pot afla mai multe solutii pentru x, care trebuiesc verificate. Timpul de lucru este din nou $O(\sqrt{p})$.

28 Atacuri din definitiile de securitate

Chosen plain-text attack. Notat si CPA sau atac pasiv, in cadrul acestui atac se presupune ca atacatorul dispune de o masina de criptare $Enc_k(x)$ dar nu dispune de masina de decriptare.

Chosen cypher-text attack. Varianta CCA1. Atacatorul are acces la o masina de decriptare, dar nu poate sa o foloseasca la discretie. Modelul real este *lunch-time attack*: ofiterul paraseste biroul pentru masa de pranz, iar spionul se infiltreaza cu acoperirea *femeie de serviciu*. Asadar atacatorul poate descifra un numar fix sau polinomial marginit de mesaje, sau mesaje pe care nu le poate alege.

Chosen cypher-text attack. Varianta CCA2, adaptive chosen cypher-text attack. Atacatorul poate descifra oricate mesaje criptate alese, in afara mesajului oficial interceptat de inamic, care il intereseaza pe atacator intr-adevar. Aceasta ipoteza este justificata de scenariul in care inamicul s-a infiltrat in rolul unui ofiter inferior. El poate folosi masina de decriptare la discretie, dar nu are acces la deciziile importante.

Observatie: *Un sistem determinist nu este CCA2-sigur.*

Demonstratie: Presupunem ca atacatorul stie ca s-a criptat unul dintre mesajele m_1 sau m_2. (Acestea pot fi "da" sau "nu", "vand" sau "cumpar", "fetita" sau "baietel"). Atacatorul intercepteaza c si cripteaza $c' = Enc_k(m_1)$. Daca $c' = c$ atunci $m = m_1$, altfel $m = m_2$. □

Observatie: *RSA pur nu este CCA2-sigur.*

Demonstratia 1: Este un sistem determinist. □

Demonstratia 2: Este o urmare a proprietatii homomorfe a lui RSA:

$$(m_1 m_2)^e \bmod N = (m_1^e \bmod N)(m_2^e \bmod N) \bmod N.$$

Presupunem ca atacatorul vrea sa decripteze $c = m^e \bmod N$. Atacatorul creeaza mesajul $c' = 2^e c \bmod N$ si decripteaza c', obtinand un mesaj m'. Acum poate obtine imediat m deoarece:

$$m' \cdot 2^{-1} \bmod N = (2^e c)^d 2^{-1} \bmod N = m.$$

Simplul non-determinism nu este suficient ca sa faca o criptare CCA2-sigura, dupa cum rezulta din urmatoarea observatie:

Observatie: *Elgamal pur nu este CCA2-sigur.*

Demonstratie: Atacatorul are de decriptat mesajul $c = (c_1, c_2) = (g^k, mh^k)$. Atacatorul creeaza mesajul $c' = (c_1, 2c_2)$, il decripteaza, si obtine mesajul m'. Din nou:

$$m' \cdot 2^{-1} = 2c_2 c_1^{-x} 2^{-1} = 2m2^{-1} = m.$$

□

Observatie: *Goldwasser-Micali pur nu este CCA2-sigur.*

Demonstratie: Atacatorul intercepteaza $c = y^b x^2 \bmod N$. El produce si descifreaza $c' = c \cdot z^2 \bmod N$, si afla $m \in \{0, 1\}$. □

29 Predicate dificile

Definitie: Fie S si T multimi de cuvinte. Fie $B : S \to \{0, 1\}$ un predicat si $f : S \to T$ o functie one-way. Se spune ca predicatul B este dificil din f, daca $B(x)$ este usor de calculat, dat x, insa $B(x)$ este greu de calculat, dat $f(x)$.

Teorema: *Fie G un grup ciclic de ordin q prim si g un generator al lui G. Fie $B_2(x)$ predicatul de paritate, definit de $B_1(x) = x \bmod 2$. Atunci B_2 este dificil din $f(x) = g^x$. Cu alte cuvinte, paritatea este dificila pentru logaritmul discret.*

Explicatia notatiei $B_1(x)$ vine abea dupa demonstratie.

Demonstratie: Fie $O(h, g)$ un oracol care calculeaza $B_1(\log_g h)$, adica ultimul bit al logaritmului discret al lui h. Iata cum putem calcula usor logaritmul discret folosind acest oracol. In programul urmator $h = g^x$, dar x nu este cunoscut.

input h;
$t = 2^{-1} \bmod q$;
$y = 0$; $z = 1$;
while $(h \neq 1)$
    $b = O(h, g)$;
    if $b = 1$ **then** $y = y + z$; $h = h/g$; **endif**
    $h = h^t \bmod q$; $z = 2z$;
return y

□

Exemplu: In F_{607} elementul $g = 64$ genereaza grupul G de ordin 101. Se poate calcula $\log_{64} 56 \bmod 101$, care este $x = 86$. Calculele in grup se fac modulo 607, numai inversul lui 2 se ia modulo 101.

Functia RSA $c = m^e \bmod N$ are urmatoarele predicate dificile:

$$\begin{aligned} B_1(m) &= m \bmod 2. \\ B_h(m) &= \begin{cases} 0, & m < \frac{N}{2}, \\ 1, & m \geq \frac{N}{2}. \end{cases} \\ B_k(m) &= m \bmod 2^k, \quad k = \log\log N. \end{aligned}$$

Notam oracolele corespunzatoare cu $O_1(c, N)$, $O_h(c, N)$ si $O_k(c, N)$. Este vorba de a afla ultimul bit al mesajului clar cunoscand mesajul criptat, etc. Observam ca intre O_1 si O_h exista urmatoarea legatura:

$$\begin{aligned} O_h(c, N) &= O_1(c \cdot 2^e \bmod N, N), \\ O_1(c, N) &= O_h(c \cdot 2^{-e} \bmod N, N). \end{aligned}$$

Predicatele sunt dificile pentru RSA. De exemplu in cazul lui O_h, daca acest predicat ar fi facil, am putea inversa foarte usor functia RSA, dupa cum urmeaza.

Declaram $y = c$, $l = 0$ si $h = N$. Atata timp cat $h - l \geq 1$ repetam urmatorul loop:

$b = O_h(y, n)$;

$y = y \cdot 2^e \bmod N$;

$m = (h + l)/2$;

if $b = 1$ **then** $l = m$ **else** $h = m$

La iesirea din loop se calculeaza $m = [h]$.

Part V
Signaturi si schimburi de cheie

30 Deffie-Hellman

Una din primele manifestari ale criptografiei cu cheie publica este schimbul de chei Diffie-Hellman. In cadrul acestui protocol, Alice si Bob comunica pe un canal public, cu securitate zero. Ei au la dispozitie un sistem simetric de criptare (Enc, Dec) foarte bun, dar la inceputul fiecarei sesiuni de comunicare simetrica ei doresc sa stabileasca de comun acord o cheie simetrica, fara ca cei care supravegheaza canalul de comunicare sa afle aceasta cheie. Alice si Bob cunosc amandoi un anumit grup ciclic G si un generator g. Eventual, la inceputul comunicarii, unul dintre ei propune perechea (G, g). Faptul ca grupul si generatorul sunt cunoscute public nu reduce gradul de securitate, intrucat securitatea se bazeaza numai pe dificultatea logaritmului discret.

Iata cum functioneaza protocolul:

Alice	Bob
Alege a, trimite g^a, calculeaza $k = (g^b)^a$.	Alege b, trimite g^b, calculeaza $k = (g^a)^b$.

Cum ambii calculeaza $k = g^{ab}$, vor avea aceeasi cheie pentru sesiunea lor de criptare simetrica, fara a fi comunicat-o in mod direct. In priviinta securitatii, putem face urmatoarele observatii:

Daca $G = (\mathbb{Z}_n, +, 0)$, protocolul este absolut nesigur in situatia in care generatorul g se cunoaste. Intr-adevar, atacatorul poate afla $h = g^{-1} \bmod n$ prin logaritmul inversului modular. Elementul h exista pentru ca conditia ca g sa fie inversabil modulo n este identica cu conditia ca g sa fie generator al lui G. Cum g^a in grupul G este de fapt $ga \bmod n$, atacatorul calculeaza $hga \bmod n = a \bmod n$, si apoi calculeaza cheia $k = (gb)a \bmod N$.

Daca G este un subgrup multiplicativ intr-un grup multiplicativ al unui inel de resturi $(\mathbb{Z}_n^\times, \cdot, 1)$ sau al unui corp finit $(\mathbb{F}_q^\times, \cdot, 1)$, protocolul este sigur daca numarul de elemente este suficient de mare. In cazul $(\mathbb{Z}_p^\times, \cdot, 1)$ putem lua p numar prim de ~ 1024 de biti.

Daca p este subgrup ciclic $G =< P >$ al unei curbe eliptice $E(\mathbb{F}_p)$, pentru acelasi nivel de securitate trebuie alese numere prime cam de 160 de biti. Asta deoarece pentru $\mathbb{F}_p^\times$ exista algoritmi subexponentiali de calcul ai logaritmului discret, dar pentru curbe eliptice nu se cunosc asemenea algoritmi.

Protocolul Diffie-Hellman in stare pura poate fi corupt prin atacul *man in the middle*. Pentru aceasta, agentul inamic Eva, il inlocuieste pe Bob in comunicarea cu Alice si o inlocuieste pe Alice in comunicarea cu Bob, asezandu-se practic intre ei. Iata schema atacului:

Alice		Eva	Eva		Bob
a	$\longrightarrow$	g^a	n	$\longrightarrow$	g^n
g^m	$\longleftarrow$	m	g^b	$\longleftarrow$	b
g^{am}		g^{am}	g^{bn}		g^{bn}

Astfel Alice calculeaza $k = g^{am}$, despre care crede ca este cheia de comunicare simetrica cu Bob, iar Bob calculeaza $k = g^{bn}$ despre care crede ca este cheia de comunicare secreta cu Alice, dar ambii comunica cu Eva. Eva poate decripta si recripta mesajele pentru a ii face sa continue comunicarea, dar aceasta comunicare nu mai are pentru ea niciun secret. De asemeni, Eva poate sabota cum doreste schimbul de informatii, de exemplu blocand anumite mesaje vitale sau modificandu-le conform propriilor ei interese.

Este clar cel putin acum importanta mesajelor cu semnatura, pentru autentificarea unui mesaj. Acest subiect a fost deja tratat tangential in contextul criptografiei simetrice, in capitolul cu functii hash respectiv MAC (message authentification code). Reluam acest subiect si pe fondul criptografiei cu cheie publica. Iata doua scheme uzuale de comunicare semnata:

Schema cu appendix. In acest caz signatura este un appendix la mesajul criptat. Procesul de calcul este urmatorul:

$$\begin{aligned} \text{mesaj} + \text{cheia secreta a lui Alice} &= \text{signatura}, \\ \text{mesaj} + \text{signatura} + \text{cheia publica a lui Alice} &= \text{da sau nu}. \end{aligned}$$

Schema cu recuperarea mesajului. In situatia in care Alice vrea sa faca un mesaj public dar vrea sa puna accent pe faptul ca mesajul este autentic, poate aplica o schema cu recuperarea mesajului. Procesul de calcul este urmatorul:

$$\begin{aligned} \text{mesaj} + \text{cheia secreta a lui Alice} &= \text{signatura}, \\ \text{signatura} + \text{cheia publica a lui Alice} &= \text{da sau nu} + \text{mesaj}. \end{aligned}$$

De exemplu RSA poate fi folosit ca algoritm de semnatura cu recuperare de mesaj. Fie d cheia secreta a lui Alice. Folosind functia de decriptare, Alice genereaza semnatura $s = m^d \bmod N$. Orice agent din public poate acum sa foloseasca cheia publica de criptare pentru a verifica autenticitatea si a recupera mesajul. El poate calcula $m = s^e \bmod N$. Dupa cum se vede, cheile publica si secreta isi schimba rolurile. Cu cheia secreta se genereaza semnatura, iar cu cheia publica semnatura poate fi verificata de catre orice persoana interesata de autenticitate. Numai eu pot semna, dar oricine se poate convinge ca semnatura este a mea.

Aceasta idee se poate imbogati cu folosirea functiilor hash. Iata schema de semnatura:

Vrem sa semnam mesajul m avand la dispozitie o functie hash publica h si o cheie secreta RSA d. Toate celelalte parti cunosc cheia publica corespunzatoare e. Calculam semnatura:

$$s = h(m)^d \bmod N,$$

si publicam mesajul semnat (m, s). Verificarea semnaturii: se calculeaza $h' = s^e \bmod N$ si $h = h(m)$. Daca $h = h'$ atunci mesajul este autentic, iar daca $h \neq h'$, mesajul nu este autentic.

31 DSA

Digital Signature Algorithm (DSA) este o schema de signatura cu appedix. Schema produce o pereche (r, s) de aproximativ 160 de biti. Se da o functie hash H, care este publica. Componenta r este functie de o cheie efemera k ce se schimba la fiecare mesaj, iar componenta s este functie de:

- mesaj,

- cheia secreta a semnatarului,

- componenta r, (deci si de cheia efemera k).

Iata pasii acestui protocol:

1. Se alege un numar prim q de aproximativ 160 biti si un numar prim p care are intre 512 si 2048 de biti, astfel incat $q|(p-1)$.

2. Se genereaza random $h < p$ si se calculeaza:

   $$g = h^{\frac{p-1}{q}} \bmod p.$$

   Se repeta procedura pana cand se obtine un $g \neq 1$. Acest g are ordin q in grupul $\mathbb{F}_p^\times$.

3. Odata ce avem (p, q, g), fiecare user genereaza cheia lui secreta x cu $0 < x < q$. Cheia publica asociata y este:

   $$y = g^x \bmod p.$$

   Secretul cheii x este asigurat de dificultatea logaritmului discret.

4. **Semnarea** mesajului m:
   - Calculeaza $h = H(m)$.
   - Alege random o cheie efemera k, $0 < k < q$.
   - Calculeaza $r = (g^k \bmod p) \bmod q$.
   - Calculeaza $s = (h + xr)k^{-1} \bmod q$.
   - Trimite (r, s) ca semnatura a lui m.
5. **Verificarea** autenticitatii mesajului semnat $[m, (r, s)]$:
   - Calculeaza $h = H(m)$.
   - Calculeaza $a = hs^{-1} \bmod q$.
   - Calculeaza $b = rs^{-1} \bmod q$.
   - Calculeaza $v = (g^a y^b \bmod p) \bmod q$, unde y este cheia publica a emitatorului.
   - Accepta semnatura daca si numai daca $v = r$.

Teorema: *Mesajele autentice sunt intotdeauna recunoscute ca atare de catre DSA.*

Demonstratie: Intr-adevar, urmatoarele egalitati au loc modulo p:

$$g^a y^b = g^{hs^{-1}} y^{rs^{-1}} = g^{hs^{-1}} g^{rxs^{-1}} = g^{(h+rx)s^{-1}} = g^k.$$

Prin urmare:

$$v = (g^a y^b \bmod p) \bmod q = (g^k \bmod p) \bmod q = r.$$

□

Exemplu: Fie $q = 13$, $p = 4q + 1 = 53$, $g = 16$. Semnatarul alege cheia secreta $x = 3$, deci cheia publica este $y = 16^3 \bmod 53 = 256 \cdot 16 \bmod 53 = 44 \cdot 16 \bmod 53 = 15$. Daca $H(m) = 5$ si cheia efemera este $k = 2$,

$$r = (g^k \bmod p) \bmod q = (16^2 \bmod 53) \bmod 13 = 44 \bmod 13 = 5,$$

$$s = (h + xr)k^{-1} \bmod q = (5 + 3 \cdot 5) \cdot 7 \bmod 13 = 10.$$

Pentru verificare, se calculeaza:

$$a = hs^{-1} \bmod q = 5 \cdot 4 \bmod 13 = 7,$$

$$b = rs^{-1} \bmod q = 5 \cdot 4 \bmod 13 = 7,$$

$$v = (g^a y^b \bmod p) \bmod q = (16^7 15^7 \bmod 53) \bmod 13 =$$

$$= (240^7 \bmod 53) \bmod 13 = (28^7 \bmod 53) \bmod 13 = 44 \bmod 13 = 5.$$

□

Exemplu: Iata un exemplu de EC-DSA, algoritmul analog cu DSA care foloseste curbe eliptice. Consideram curba eliptica data de $y^2 = x^3 + x + 3$ pest corpul $\mathbb{F}_{199}$. Curba are $q = 197$ de elemente, deci este un grup ciclic deoarece 197 este prim. Se poate lua ca generator punctul $P = (1, 76)$.

Consideram cheia privata $x = 29$. Cheia publica este $Y = [x]P = [29](1, 76) = (113, 191)$.

Presupunem ca userul vrea sa semneze un mesaj cu valoarea $H(m) = 68$. El produce cheia efemera $k = 153$ si calculeaza:

$$r = \pi_x([k]P) = \pi_x([153](1, 76)) = \pi_x(185, 35) = 185.$$

Apoi calculeaza:

$$s = (H(m) + xr)k^{-1} \bmod q = (68 + 29 \cdot 185)153^{-1} \bmod 197 = 78.$$

Signatura este $(r, s) = (185, 78)$. Verificarea decurge in modul urmator:

$$a = H(m)s^{-1} \bmod q = 68 \cdot 78^{-1} \bmod 197 = 112,$$

$$b = rs^{-1} \bmod q = 185 \cdot 78^{-1} \bmod 197 = 15,$$

$$Z = [a]P + [b]Y = [112](1, 76) + [15](113, 191) = (111, 60) + (122, 140) = (185, 35),$$

$$r = \pi_x(Z) = 185.$$

□

32 MQV

Schimbul de cheie MQV (Menezes–Qu–Vanstone) este o generalizare a schimbului de chei Diffie-Hellman. In cele ce urmeaza fie G un grup ciclic generat de un element public g. Acest grup poate fi si o curba eliptica, asa cum este implementat protocolul la NIST. Alice si Bob au perechi (cheie publica, cheie secreta) de lunga durata, de forma:

$$(A = g^a, a) \;\; ; \;\; (B = g^b, b).$$

Pentru o sedinta de schimb de cheie, ei genereaza niste chei efemere:

$$(C = g^c, c) \;\; ; \;\; (D = g^d, d).$$

Alice ii trimite lui Bob C, iar Bob ii trimite lui Alice D. Fie L definit de:

$$L = \left\lceil \frac{[\log_2 |G|] + 1}{2} \right\rceil.$$

Alice cunoaste acum valorile A, B, C, D, a, c si calculeaza:

$$\begin{aligned} s_A &= 2^L + (C \bmod 2^L), \\ t_A &= 2^L + (D \bmod 2^L), \\ h_A &= c + s_A a, \\ P_A &= (DB^{t_A})^{h_A}. \end{aligned}$$

Bob cunoaste la randul lui valorile A, B, C, D, b, d si calculeaza simetric:

$$\begin{aligned} s_B &= 2^L + (D \bmod 2^L), \\ t_B &= 2^L + (C \bmod 2^L), \\ h_B &= d + s_B b, \\ P_B &= (CA^{t_B})^{h_B}. \end{aligned}$$

Teorema: *In conditiile de mai sus, $P_A = P_B$ este secretul impartasit.*

Demonstratie: Se observa ca prin definitie, $s_A = t_B = x$ si $s_B = t_A = y$. In randurile de mai jos, $\log_g$ este logaritmul discret in raport cu generatorul g.

$$\log_g P_A = \log_g((DB^y)^{h_A}) = (d + yb)(c + xa).$$

$$\log_g P_B = \log_g((CA^x)^{h_b}) = (c + xa)(d + yb).$$

□

33 OAEP-RSA

OAEP-RSA (Optymized asymmetric encryption padding for RSA) este un protocol mecanizat care contine primitiva RSA si are ca scop comunicarea CCA2 sigura. Iata principiul OAEP:

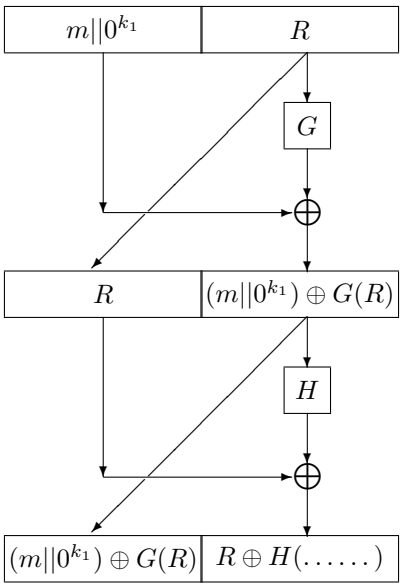
Fie $f : \{0,1\}^k \to \{0,1\}^k$ o functie bijectiva (permutatie) trap-door si one-way, de exemplu criptarea RSA data de $f(m) = m^e \bmod N$. Fie k_0, k_1 numere naturale astfel incat 2^{k_0} si 2^{k_1} sunt prea mari pentru a fi tratate prin forta bruta, de exemplu $k_0, k_1 \geq 128$. Fie $n = k - k_0 - k_1$. Vom cripta mesaje (blocuri) de lungime n. In particular $n + k_1 = k - k_0$.

Se construiesc doua functii hash G si H dupa cum urmeaza:

$$G : \{0,1\}^{k_0} \to \{0,1\}^{n+k_1},$$

$$H : \{0,1\}^{n+k_1} \to \{0,1\}^{k_0}.$$

Protocolul consta din doua runde Feistel asimetrice. Aici R este un cuvant binar random de lungime k_0.



Functia f (respectiv RSA) se aplica pe rezultatul acestei retele Feistel. Asadar criptarea este:

$$Enc(m) = f([(m||0^{k_1}) \oplus G(R))]||[R \oplus H((m||0^{k_1}) \oplus G(R))]).$$

Decriptarea se face in modul urmator. Se aplica f^{-1} pe textul criptat c, se obtine:

$$f^{-1}(c) = T||(R \oplus H(T)),$$

unde $T = (m||0^{k_1}) \oplus G(R)$. Calculam $H(T)$ si deducem $R = (R \oplus H(T)) \oplus H(T)$. Acum putem calcula $G(R)$ si deducem m din $m||0^{k_1} = [(m||0^{k_1}) \oplus G(R)] \oplus G(R)$. □

Teorema: *RSA-OAEP este semantic sigura impotriva atacurilor CCA2.*

Part VI
Protocoale avansate

34 INS secrete sharing

Secrete sharing (secret partitionat) inseamna impartirea unor elemente criptografice mai multor agenti (persoane, computere, organizatii) cu scopul de a putea descifra un anumit secret numai impreuna si cu acordul tuturor.

Exemplul 1: Fie s parola secreta pentru folosirea unei arme nucleare. Presupunem ca secretul s este partitionat intre 4 agenti care au roluri, respectiv importanta, diferita. Astfel, detinatori ai unor parti din secret pot fi Presedintele P, Vicepresedintele V, Secretarul Apararii (Secretary of Defence) S si Generalul G. Vrem ca urmatoarele multimi minimale sa poata lansa bomba atomica:

$$\{P, G\} \ ; \ \{V, S, G\},$$

dar nicio alta submultime. Vom nota cu s_P, s_V, s_S si s_G secretul partial primit de fiecare agent. □

Prin **structura de acces** intelegem o colectie $\Gamma \subset \mathcal{P}(P)$cu proprietatile ca $P \in \Gamma$ si pentru orice $A \in \Gamma$, daca $A \subset B$, atunci $B \in \Gamma$. **Schema de partitionare a secretului** este formata din doi algoritmi numiti **Share** si **Recombine**:

Share(s, Γ) repartizeaza fiecarei persoane X un secret propriu s_X.

Recombine considera $H = \{s_X \mid X \in \mathcal{O}\}$. Daca $\mathcal{O} \in \Gamma$ il calculeaza pe s, altfel da un mesaj de eroare.

In exemplul cu Presedintele,

$$\Gamma = \{PG, VSG, PGV, PGS, PVGS\},$$

iar PG si VSG sunt multimile minimale.

Exemplul 2: Patru directori A, B, C si D au drepturi egale. De comun acord, ei stabilesc regula ca cel putin doi dintre ei trebuie sa fie de fata cand se deschide fisierul cu contracte. In acest caz:

$$\Gamma = \{AB, AC, AD, BC, BD, CD, ABC, ABD, ACD, BCD, ABCD\},$$

unde $\{AB, AC, AD, BC, BD, CD\}$ sunt multimile minimale.

Schema Ito-Niszizeki-Saito. Fie $m(\Gamma)$ multimea submultimilor minimale. Pentru orice $\mathcal{O} \in m(\Gamma)$, daca $k = |\mathcal{O}|$, generam secrete $s_1, \ldots, s_k$ astfel incat:

$$s_1 \oplus s_2 \oplus \cdots \oplus s_k = s.$$

In practica putem genera $k-1$ astfel de secrete random, iar pe s_k il calculam din:

$$s_k = s_1 \oplus \cdots \oplus s_{k-1} \oplus s.$$

Exemplul 1: In acest caz se fac descompunerile:

$$s = s_1 \oplus s_2 = s_3 \oplus s_4 \oplus s_5.$$

Partajarea se face dupa urmatoarea schema:

$$\begin{aligned} s_P &= s_1, \\ s_V &= s_3, \\ s_S &= s_4, \\ s_G &= s_2 || s_5. \end{aligned}$$

algoritmul recombine trebuie intai sa constate prezenta unei multimi din schema de acces, apoi sa reconstituie secretul din secretele partiale ale unei asemenea submultimi.

Exemplul 2: In acest caz se calculeaza 6 descompuneri:

$$s = s_{AB} \oplus t_{AB} = s_{AC} \oplus t_{AC} = s_{AD} \oplus t_{AD} = s_{BC} \oplus t_{BC} = s_{BD} \oplus t_{BD} = s_{CD} \oplus t_{CD},$$

iar secretele care se partajeaza sunt:

$$\begin{aligned} s_A &= s_{AB}||s_{AC}||s_{AD}, \\ s_B &= t_{AB}||s_{BC}||s_{BD}, \\ s_C &= t_{AC}||t_{BC}||s_{CD}, \\ s_D &= t_{AD}||t_{BD}||t_{CD}. \end{aligned}$$

35 RSS secrete sharing

Schema RSS (Replicated Secret Sharing) se realizeaza in urmatorul mod:

1. Fie A_1, A_2, ..., A_t multimile maximale care *nu* sunt in Γ. Asta inseamna ca orice multime $B \neq A_i$ astfel incat $A_i \subset B$ este in Γ.
2. Fie B_1, ..., B_t complementele lui A_1, ..., A_t.
3. Se alege un secret s si se descompune s ca suma binara de t elemente:
$$s = s_1 \oplus s_2 \oplus \cdots \oplus s_t.$$
4. Persoana P primeste secretul s_i daca si numai daca $P \in B_i$.

Exemplul 1: In acest caz multimile maximale slabe sunt:

$$A_1 = \{P, V, S\} \quad , \quad A_2 = \{V, G\} \quad , \quad A_3 = \{S, G\}.$$

Complementele lor sunt:

$$B_1 = \{G\} \quad , \quad B_2 = \{S, P\} \quad , \quad B_3 = \{P, V\}.$$

Descompunerea va fi:

$$s = s_1 \oplus s_2 \oplus s_3,$$

iar distributia va fi:

$$\begin{aligned} s_P &= s_2||s_3, \\ s_V &= s_3, \\ s_S &= s_2, \\ s_G &= s_1. \end{aligned}$$

Exemplul 2: In acest caz multimile maximale slabe sunt:

$$A_1 = \{A\} \quad , \quad A_2 = \{B\} \quad , \quad A_3 = \{C\} \quad , \quad A_4 = \{D\},$$

iar complementele lor sunt:

$$B_1 = \{B, C, D\} \quad , \quad A_2 = \{A, C, D\} \quad , \quad A_3 = \{A, B, D\} \quad , \quad A_4 = \{A, B, C\}.$$

Descompunerea va fi:

$$s = s_1 \oplus s_2 \oplus s_3 \oplus s_3,$$

iar distributia va fi:

$$\begin{aligned} s_A &= s_2||s_3||s_4, \\ s_B &= s_1||s_3||s_4, \\ s_C &= s_1||s_2||s_4, \\ s_D &= s_1||s_2||s_3. \end{aligned}$$

36 Shamir secrete sharing

Schemele de secrete sharing din sectiunile precedente sunt foarte ineficiente, mai ales in priviinta algoritmului de recombinare. Schema lui Shamir, pe care o prezint acum, este o schema algebrica eficienta pentru un caz extrem, in care schemele anterioare se dovedesc deosebit de ineficiente: cazul in care *fiecare submultime de t persoane din n nu pot reconstrui secretul, dar orice submultime de $t+1$ persoane din n poate reconstrui secretul si dobandeste acces.*

Iata schema lui Shamir:

1. Se alege un corp $\mathbb{F}_q$ cu $q > n$.
2. Secretul partajat este un element $s \in \mathbb{F}_q$ care se alege la intamplare. Se mai aleg la intamplare t elemente, nu neaparat distincte, $f_1, \ldots, f_t \in \mathbb{F}_q$, si se considera polinomul:
$$f(X) = s + f_1 X + \cdots + f_t X^t \in \mathbb{F}_q[X].$$
3. Fiecare persoana primeste o alta eticheta $x_i \in \mathbb{F}_q$. Persoana i primeste $s_i = (x_i, f(x_i))$.

Teorema: *In situatia descrisa mai sus, orice submultime de $t+1$ persoane poate reconstitui secretul $s = f(0)$, pe cand orice submultime de t persoane nu il poate reconstitui.*

Demonstratia 1: Alegem $t+1$ persoane. Fiecare vine cu o ecuatie, mai precis persoana etichetata x_i poarta cu sine ecuatia:
$$s + f_1 x_i + f_2 x_i^2 + \cdots + f_t x_i^t = f(x_i).$$
Cele $t+1$ ecuatii formeaza un sistem de ecuatii liniare cu $t+1$ necunoscute. Determinantul matricii:
$$(x_i^j)_{i=1,\ldots,t+1}^{j=0,\ldots,t},$$
este asa-zisul determinant Vandermonde, cu valoarea:
$$\prod_{i>k}(x_i - x_k) \neq 0.$$
Asadar sistemul este rezolvabil, si are o unica solutie $(s, f_1, \ldots, f_t)$. Componenta s este secretul cautat. Daca numarul de ecuatii este mai mic decat $t+1$, solutia nu este unica, deci nu putem gasi secretul s. □

Demonstratia 2: Privim problema ca pe o problema de interpolare. In cazul acesta trebuie gasit un polinom de grad t care trece prin punctele $(x_i, f(x_i))$. punctele sunt diferite, pentru ca au abscise diferite. In acest caz,
$$f(X) = \sum_{i=0}^{t} f(x_i) \prod_{k=0, k\neq i}^{t} \frac{X - x_k}{x_i - x_k} = \sum_{i=0}^{t} f(x_i)\delta_i(X).$$
Asadar secretul se poate obtine direct ca:
$$s = f(0) = \sum_{i=0}^{t} f(x_i)\delta_i(0).$$
□

37 Angajamente

Angajamente (commitments) se refera la punerea unui pariu criptat, care sa nu mai poata fi schimbat sau reinterpretat dupa anuntarea rezultatului oficial. Vom descrie acest terme printr-un exemplu sugestiv:

Bob si Alice joaca *piatra-hartie-foarfeca* prin telefon. Se foloseste o functie hash publica, notata H. Iata cum arata o runda de joc. Mai jos, R_A este un string random produs in momentul transmisiei.

$$\begin{aligned} A \to B \qquad & h_A = H(R_A||HARTIE), \\ B \to A \qquad & FOARFECE, \\ A \to B \qquad & R_A||HARTIE. \end{aligned}$$

Acum Bob poate verifica, cu ajutorul functiei hash, ca Alice a jucat corect cand a pariat. Daca Alice ar vrea sa triseza, ar trebui sa gaseasca un string R'_A astfel incat:

$$H(R_A||HARTIE) = H(R'_A||PIATRA).$$

Acest lucru este greu de facut daca functia H este rezistenta la a doua preimagine.

Definitie: O schema de angajament este **binding** (il leaga pe adversar de agajamentul lui) daca niciun adversar nu poate castiga urmatorul joc:

- Adversarul produce c, x, r astfel incat $c = C(x, r)$.

- Adversarul produce $x' \neq x$ si r' astfel incat $C(x, r) = C(x', r')$.

Definitie: O schema de angajament este **concealing** (sigura) daca niciun adversar nu poate castiga urmatorul joc cu probabilitate mai mare decat $1/2$:

- Adversarul produce cuvinte $x_0 \neq x_1$ cu $|x_0| = |x_1|$.

- Autoritatea genereaza r random si $b \in \{0, 1\}$.

- Autoritatea calculeaza $c = C(x_b, r)$ pe care il trimite adversarului.

- Adversarul ghiceste b.

Teorema: *Nicio schema nu poate fi absolut binding si absolut concealing.*

Demonstratie: Daca schema este absolut binding, aplicatia de la angajamente la valori angajate va fi injectiva. Prin urmare schema este determinista, si nu poate fi absolut concealing. □

In acest context, observam ca schema $H(R||M)$ introdusa mai sus este computationally binding si concealing in limitele date de teoria informatiei (se spune information-theoretically concealing). Acestea sunt niste relativizari acceptabile ale notiunilor introduse mai sus.

In practica se folosesc urmatoarele scheme bazate pe dificultatea logaritmului discret. Fie G un grup ciclic de ordin q prim, $G = \langle g \rangle$ cu $g \in \mathbb{F}_p^\times$, si un $h \in G$ cu $\log_g h$ necunoscut de catre userul care face un angajament. Valoarea angajata este un $x \bmod q$.

Schema exponentiala este $B(x) = g^x$.

Schema lui Pedersen este $B_a(x) = h^x g^a$, unde a este ales random.

Schema Elgamal este $E_a(x) = (g^a, xh^a)$.

Observatie: $B(x)$ este information-theoretically concealing, dar este injectiva.

Observatie: $E_a(x)$ este information-theoretically binding si computationally concealing.

Observatie: $B_a(x)$ este information-theoretically binding si computationally concealing.

Observatie: Schemele B si B_a au proprietati homomorfice:

$$B(x_1)B(x_2) = B(x_1 + x_2),$$

$$B_{a_1}(x_1)B_{a_2}(x_2) = B_{a_1+a_2}(x_1 + x_2).$$

Acest lucru este avantajos in anumite contexte, si va fi exploatat in votul electronic.

38 Transfer partial (oblivious transfer)

Ca si angajamentele criptate, transferurile partiale se folosesc pentru a crea o legatura intre doua parti care nu au incredere una in cealalta. Protocolul trebuie sa respecte urmatoarele conditii:

Un Emitator trimite doua mesaje m_0 si m_1 in sistem. Receptorul comunica sistemului un $b \in \{0,1\}$ ales la intamplare, si obtine de la sistem mesajul m_b. Receptorul nu trebuie sa afle niciodata continutul mesajului m_{1-b} iar Emitatorul nu trebuie sa afle niciodata b. In acest fel intre cele doua parti apare o relatie de complicitate, in ciuda situatiei de neincredere in care se afla.

Inainte de descrierea concreta a unui sistem de transfer partial, reluam sistemul de criptare Elgamal intr-o varianta imbogatita cu o functie hash. Fie $G = \langle g \rangle$ un grup ciclic de ordin q, o pereche (cheie publica, cheie secreta) $= (h = g^x, x)$, si $H : G \to \{0,1\}^n$ o functie hash, unde $n = \log q$. Pentru mesaje m de lungime n, Bob alege o cheie efemera k si calculeaza perechea:

$$(c_1, c_2) = (g^k, m \oplus H(h^k)).$$

Pentru decriptare Alice calculeaza:

$$c_2 \oplus H(c_1^x) = m \oplus H(h^k) \oplus H(g^{kx}) = m.$$

Ideea protocolului este urmatoarea: Receptorul are doua chei publice, dar cunoaste numai una din cele doua chei secrete corespunzatoare. Emitatorul codifica cele doua mesaje, fiecare cu una dintre cheile publice, si le trimite. Receptorul poate decripta numai un mesaj, iar emitatorul nu stie care dintre mesaje a fost decriptat.

Concret, implementarea se poate face in modul urmator:

- Emitatorul alege $c \in G$ si nu cunoaste $\log_g c$. Trimite c receptorului.

- Receptorul alege $x \in \mathbb{Z}_q$ random. El calculeaza cheile publice:

$$h_b = g^x \quad , \quad h_{1-b} = c \cdot h_b^{-1}.$$

Ca urmare, emitatorul cunoaste cheia secreta x corespunzatoare lui h_b dar nu si cheia secreta corespunzatoare lui h_{1-b}. El trimite emitatorului cheia h_0.

- Emitatorul calculeaza

$$h_1 = c \cdot h_0^{-1}.$$

- Emitatorul alege $k \in \mathbb{Z}_q$ cheie efemera si calculeaza $c_1 = g^k$. El mai calculeaza:

$$e_0 = m_0 \oplus H(h_0^k),$$

$$e_1 = m_1 \oplus H(h_1^k).$$

- Emitatorul transmite receptorului c_1, e_0 si e_1.

- Receptorul poate descifra doar $m_b = e_b \oplus H(c_1^x)$. Emitatorul nu stie care mesaj a fost descifrat.

39 Zero Knowledge Proofs

Protocoalele Zero Knowledge sunt scenarii de partajare a secretelor intre doua persoane: un prover numit Peggy si un verifier numit Victor. Peggy vrea sa il convinga pe Victor ca stie un anumit lucru, fara sa i-l dezvaluie, sau dezvaluind numai parti infime din secret. Victor trebuie sa verifice in mod convingator ca Peggy detine secretul, fara sa il afle. Numele Zero Knowledge vine de la aceasta ignoranta a lui Victor, care trebuie garantata atat in timpul desfasurarii protocolului, cat si dupa aceea.

Problema Izomorfismului de Grafuri a generat exemplul clasic de Zero Knowledge Proof. Un graf $G = (V, E)$ este format dintr-o multime de varfuri V si o multime de muchii $E \subset V \times V$. Fie grafuri $G_i = (V_i, E_i)$, cu $i = 0, 1$. O functie $\varphi : V_1 \to V_2$ se numeste *izomorfism de grafuri* daca si numai daca:

1. φ este bijectie.
2. $xy \in E_0$ daca si numai daca $\varphi(x)\varphi(y) \in E_1$.

In exemplul clasic de Zero Knowledge Proof avem urmatoarea situatie:

- Grafurile izomorfe $G_0 \simeq G_1$ sunt publice. Este vorba de grafuri mari, de cel putin 10^4 varfuri.

- Peggy stie un izomorfism secret $\varphi : G_0 \to G_1$. Ea vrea sa il convinga pe Victor ca are un asemenea izomorfism, fara sa i-l dezvaluie.

- Peggy alege la intamplare un $i \in \{0, 1\}$ si aplica o permutare secreta si random ψ pe varfurile lui G_i. In acest mod obtine un alt graf H. Asadar Peggy cunoaste acum urmatoarele izomorfisme:

$$\begin{aligned} \varphi : &\quad G_0 \quad \longrightarrow G_1, \\ \psi : &\quad G_i \quad \longrightarrow H, \\ \alpha : &\quad G_{1-i} \quad \longrightarrow H. \end{aligned}$$

In cazul in care $i = 1$, $\alpha = \psi \circ \varphi$, iar daca $i = 0$, $\alpha = \psi \circ \varphi^{-1}$. Ambele sunt usor de evaluat, prin cautare binara, deoarece izomorfismele sunt niste liste sau dictionare finite. Peggy publica H.

- Victor lanseaza o provocare: alege $b \in \{0, 1\}$ si cere izomorfismul grafurilor G_b si H. Peggy trimite ψ sau α.

Daca Peggy nu ar cunoaste φ, ar putea sa raspunda corect numai cu probabilitate de $1/2$. Prin repetarea procedurii, minciuna va iesi la iveala, deoarece raspuns corect de n ori se intampla cu probabilitate de $(1/2)^n$. □

Protocolul de identificare al lui Schnorr. Se da un grup ciclic de $G = g$ de ordin q si un element public $y \in G$. Secretul lui Peggy este logaritmul discret $x = \log_g y$. Protocolul are urmatorii pasi:

1. Peggy alege k random si ii trimite lui Victor $r = g^k$.
2. Victor ii trimite lui Peggy o provocare (challenge) e.
3. Peggy ii trimite lui Victor $s = k + xe$. Ea poate face acest lucru, deoarece numai ea cunoaste x si k.
4. Victor calculeaza $g^s y^{-e}$ si verifica daca a obtinut rezultatul r.

Teorema: *Daca Peggy cunoaste $x = \log_g y$, protocolul da un rezultat pozitiv. Victor il regaseste pe r fara a il afla pe x. Daca Peggy triseaza, poate obtine rezultatul pozitiv doar cu probabilitate de $1/q$.*

Demonstratie: Intr-adevar,

$$g^s y^{-e} = g^{k+xe-xe} = g^k = r.$$

□

Observatie: Trebuie evitata alegerea aceluiasi k efemer, deoarece in acest caz Victor poate sa il afle pe x. Din:

$$r = g^s y^{-e} = g^{s'} y^{-e'},$$

rezulta:

$$s + x(-e) = s' + x(-e') \bmod q,$$

$$x = (s - s')(e - e')^{-1} \bmod q.$$

□

Zero Knowledge cu angajamente Pedersen. Fie $G = \langle g \rangle$ un grup ciclic de ordin prim q, $h \in G$ cu $\log_g h$ necunoscut. Pentru a angaja o valoare x, se alege un a random si se publica angajamentul:

$$B_a(x) = h^x g^a.$$

Un caz particular important este $x \in \{-1, 1\}$. Persoana acre face angajamentul, trebuie sa demonstreze ca

$$x \in \{-1, 1\}$$

fara a dezvalui in nicio masura valoarea lui x. Iata derularea protocolului:

1. Peggy alege random d, r, w mod$q$.
2. Peggy publica $B_a(x)$, α_1, α_2, unde:

$$\alpha_1 = \begin{cases} g^r (B_a(x)h^{-1})^{-d} = g^{r-ad}, & x = 1, \\ g^w, & x = -1. \end{cases}$$

$$\alpha_2 = \begin{cases} g^w, & x = 1, \\ g^r (B_a(x)h^{+1})^{-d} = g^{r-ad}, & x = -1. \end{cases}$$

3. Victor trimite o provocare random c. (challenge)
4. Peggy calculeaza:

$$\begin{aligned} d' &= c - d, \\ r' &= w + ad'. \end{aligned}$$

   si ii trimite lui Victor:

$$(d_1, d_2, r_1, r_2) = \begin{cases} (d, d', r, r'), & x = 1, \\ (d', d, r', r), & x = -1. \end{cases}$$

5. Victor verifica urmatoarele egalitati:

$$\begin{aligned} c &= d_1 + d_2, \\ g^{r_1} &= \alpha_1 (B_a(x)h)^{d_1}, \\ g^{r_2} &= \alpha_2 (B_a(x)h^{-1})^{d_2}. \end{aligned}$$

6. Victor accepta angajamentul facut de Peggy daca si numai daca toate cele trei egalitati se dovedesc corecte.

Teorema: *In cazul in care Peggy angajeaza o valoare $x \in \{-1, 1\}$, protocolul de mai sus functioneaza cu succes, fara ca Victor sa afle valoarea angajata.*

Demonstratie: Egalitatea $c = d_1 + d_2$ se verifica automat. Daca $x = 1$, atunci:

$$\alpha_1 (B_a(x)h)^{d_1} = g^r (B_a(x)h^{-1})^{-d} (B_a(x)h)^d = g^r = g^{r_1},$$

$$\alpha_2 (B_a(x)h^{-1})^{d_2} = g^w (B_a(x)h^{-1})^{d'} = g^{w+ad'} = g^{r'} = g^{r_2}.$$

Daca $x = -1$ ambele relatii se verifica analog. Intr-adevar, Victor nu detine suficienta informatie pentru a afla daca $x = 1$ sau $x = -1$. □

40 Vot electronic

Prezentam aici sistemul de vot pentru un referendum. Votul exprimat este un angajament $v \in \{-1, 1\}$. Presupunem ca sunt m votanti indexati cu litera j. Corectitudinea votului si rezultatul lui se monitorizeaza din n centre de numerotare independente, indexate cu litera i. Aceste centre de numerotare supervizeaza intregul proces de vot, fara o anumita dependenta teritoriala.

Lista de features care trebuiesc verificate pentru implementare, este urmatoarea:

- Numai persoane autorizate voteaza.

- Fiecare persoana cu drept de vot, voteaza cel mult o data.

- Niciun votant nu poate afla cum a votat altcineva.

- Niciun vot nu se poate duplica.

- Rezultatul se calculeaza corect.

- Toti participantii pot verifica corectitudinea procesului.

- Protocolul functioneaza chiar daca se incearca trisarea lui.

Setup. Fie $G = \langle g \rangle$ grup ciclic de ordin q prim, $h \in G$, $h = g^x$, nimeni nu cunoaste x. Fiecare centru de numerotare i are o cheie publica E_i. Fiecare votant j are o cheie publica de signatura. Se da o functie hash publica H.

Votare. Fiecare alegator j:

1. Alege optiunea de vot $v_j \in \{-1, 1\}$.
2. Alege random $a_j \in \mathbb{Z}_q$.
3. Publica buletinul de vot criptat ca angajament Pedersen, $B_j = B_{a_j}(v_j)$.
4. Publica un certificat $c_j = H(\alpha_1 || \alpha_2 || B_{a_j}(v_j))$ care arata ca $v_j \in \{-1, 1\}$ fara a dezvalui valoarea lui v_j.
5. Perechea (B_j, c) este semnata cu algoritmul public de signatura.

Distribuirea voturilor. Se foloseste schema de partajare a secretului a lui Shamir pentru a distribui valorile a_j si v_j centrelor independente de numarare a voturilor.

1. Fiecare alegator j alege urmatoarele polinoame random modulo q. Gradul t al polinoamelor satisface $t < n$, unde n este numarul centrelor de numerotare.

$$\begin{aligned} R_j(x) &= v_j + r_{1,j}x + \cdots + r_{t,j}x^t, \\ S_j(x) &= a_j + s_{1,j}x + \cdots + s_{t,j}x^t. \end{aligned}$$

2. Pentru orice centru i, cu $1 \leq i \leq n$, alegatorul j calculeaza perechea:

$$(u_{i,j}, w_{i,j}) = (R_j(i), S_j(i)) \bmod q.$$

3. Alegatorul j cripteaza perechea $(u_{i,j}, w_{i,j})$ folosind cheia publica E_i a centrului de numerotare, si o trimite centrului i.

4. Alegatorul j publica angajamentele lui relativ la polinoamele R_j si S_j sub forma unor angajamente Pedersen:

$$B_{l,j} = B_{s_{l,j}}(r_{l,j})$$

pentru toti l cu $1 \leq l \leq t$.

Verificarea consistentei. Fiecare centru de numarare i trebuie sa verifice ca valoarea lui $(u_{i,j}, w_{i,j})$ primit de la alegatorul j este consistent cu alegerile facute de alegator. In acest scop se aplica proprietatile homomorfe ale angajamentului Pedersen. Se calculeaza si se verifica:

$$B_j \prod_{l=1}^{t} {B_{l,j}}^{i^l} = B_{a_j}(v_j) \prod_{l=1}^{t} B_{s_{l,j}}(r_{l,j})^{i^l} = h^{v_j} g^{a_j} \prod_{l=1}^{t} \left(h^{r_{l,j}} g^{s_{l,j}}\right)^{i^l} =$$

$$= h^{\left(v_j + \sum_{l=1}^{t} r_{l,j} i^l\right)} g^{\left(a_j + \sum_{l=1}^{t} s_{l,j} i^l\right)} = h^{u_{i,j}} g^{w_{i,j}}.$$

Numararea voturilor. Fiecare din cele n centre de numarare publica suma voturilor primite:

$$T_i = \sum_{j=1}^{m} u_{i,j},$$

si suma criptarilor de valori alese random:

$$A_i = \sum_{j=1}^{m} w_{i,j}.$$

Orice centru de numarare si orice alegator poate verifica consistenta acestor informatii folosind cealalta proprietate homomorfica a angajamentului Pedersen:

$$\prod_{j=1}^{m} \left(B_j \prod_{l=1}^{t} {B_{l,j}}^{j^l}\right) = \prod_{j=1}^{m} h^{u_{i,j}} g^{w_{i,j}} = h^{T_i} g^{A_i}.$$

Fiecare alegator si centru de votare poate poate calcula rezultatul la sfarsitul scrutinului considerand $t+1$ dintre valorile T_i si rezolvand un sistem liniar de ecuatii sau o problema de interpolare:

$$T_i = \sum_{j=1}^{m} u_{i,j} = \sum_{j=1}^{m} R_j(i) =$$

$$= \left(\sum_{j=1}^{m} v_j\right) + \left(\sum_{j=1}^{m} r_{1,j}\right) i + \cdots + \left(\sum_{j=1}^{m} r_{t,j}\right) i^t$$

Asadar sistemul contine $t+1$ ecuatii de forma:

$$V + W_1 i + \cdots + W_t i^t = T_i.$$

Sistemul se rezolva. In particular se afla $V = \sum_{j=1}^{m} v_j$, de unde se deduce cati de $+1$ si cati de -1 s-au votat, deoarece numarul de persoane care au participat la vot este cunoscut.

□

41 Problema milionarilor

Alice, Bob si Charles vor sa afle care dintre ei castiga cel mai mult, fara a destainui cat castiga fiecare. Iata cum se poate rezolva aceasta problema:

Etapa I. Alice alege random un numar r, foarte mare si cu cifre random. La acest numar aduna salariul ei a. Trimite lui Bob $r+a$, care adauga salarriul lui b. Acesta ii trimite lui Charles suma

$r + a + b$, Charles adauga si el salariul lui c, si trimite suma totala $r + a + b + c$ lui Alice. Alice anunta acum public:

$$m = \frac{a + b + c}{3}.$$

Toti cei care au un salariu $\leq m$, anunta acest lucru. Daca toti trei fac acest anunt, atunci aveau de la bun inceput salarii egale, si protocolul se incheie. Daca doi dintre ei fac acest anunt, inseamna ca al treilea are cel mai mare salariu, si protocolul se incheie. In cazul in care numai unul dintre ei face acest anunt, trebuie sa se decida care din ceilalti doi are salariul cel mai mare. Acest lucru se decide in Etapa II.

Se poate porni cu un numar arbitrar de participanti. In acest caz se repeta Etapa I pana cand problema se rezolva, sau pana cand raman doi participanti.

Etapa II. Presupunem ca Alice si Bob vor sa isi compare salariul fara ca vreunul dintre ei sa destainuie valoarea lui exacta. Ei gasesc o unitate de masura convenabila astfel incat lefurile lor sa se exprime prin numere a si b cu $0 \leq a, b \leq 100$. Fie B cheia publica a lui Bob si B' cheia lui secreta. Alice alege un x arbitrar si calculeaza $c = Enc_B(x)$. Alice trimite $d = c - a$ lui Bob.

Bob calculeaza $y_i = Dec_{B'}(d + i)$ cu $i = 0, 1, \ldots, 101$.

Bob calculeaza $z_i = f(y_i)$ cu $i = 0, 1, \ldots, 101$. Aici f este o functie hash.

Daca exista $i > j$ cu $|z_i - z_j| \leq 1$, se reia protocolul. Pentru o buna alegere a lui x, acest lucru nu se intampla.

Bob ii trimite lui Alice sirul:

$$z_0, z_1, \ldots, z_b, z_{b+1} + 1, z_{b+2} + 1 \ldots, z_{101} + 1.$$

Alice calculeaza $f(x)$ si $f(x+1)$ si le cauta in sir. Daca gaseste $f(x)$, Bob castiga mai mult. Daca gaseste $f(x + 1)$, Alice castiga mai mult.

42 Secretul de stat

Iata o situatie ipotetica. De la SRI Bucuresti pleaca n grupe de cate 3 agenti la Cluj. Ei duc la Cluj o informatie sau un obiect strict secret. Un singur agent din cei $3n$ are secretul. Fiecare agent a fost instruit sa se comporte cu colegii sai din grupa de 3 ca si cum el este cel care are secretul. In general, fiecare dintre ei stie ca nu are secretul, dar presupune ca unul dintre colegi il are. Ei se apara reciproc.

SRI Cluj vrea sa afle repede in ce grupa este persoana care detine secretul. Toate grupele sunt asezate la mese, fiecare grupa primeste o masa rotunda. In timp ce servesc cafeaua de bun-venit, fiecare agent stabileste un bit secret cu cei doi colegi de masa. Astfel, la masa la care stau agentii A, B si C se stabilesc bitii b_{AB}, b_{BC} si b_{AC}. Fiecare agent scrie pe un servetel suma booleana a bitilor pe care ii are, astfel:

$$\begin{aligned} A: &\quad b_{AB} \oplus b_{AC} = x, \\ B: &\quad b_{AB} \oplus b_{BC} = y, \\ C: &\quad b_{AC} \oplus b_{BC} = z. \end{aligned}$$

Chelnerul strangand servetelele, calculeaza

$$x \oplus y \oplus z = 0,$$

si trage concluzia ca secretul nu se afla la masa respectiva. Unicul agent care are secretul va scrie:

$$A: \quad b_{AB} \oplus b_{AC} \oplus 1.$$

La masa lui suma va fi:

$$x \oplus y \oplus z = 1,$$

si chelnerul afla ca secretul de stat se afla la masa respectiva. □

43 Circuite booleene

In cadrul acestui protocol, Alice si Bob vor sa evalueze o functie f pentru un input a al lui Alice si un input b al lui Bob, fara sa-si destainuie unul altuia valoarea lui a si b. Ei trebuie sa aiba incredere in calcul si in rezultatul lui.

Cum orice input se poate trasforma intr-o pereche de cuvinte binare si orice functie numerica se poate calcula folosind un circuit boolean, ei cripteaza circuitul folosind metoda circuitelor deformate a lui Yao. Circuitul nedeformat, care calculeaza functia $f(x, y)$, este cunoscut de la inceput ambelor parti.

Schema generala este urmatoarea:

- Alice cripteaza circuitul si inputul ei a.

- Bob cripteaza inputul lui b folosind ajutorul lui Alice prin protocolul *transfer partial*.

- Bob evalueaza circuitul si obtine rezultatul criptat.

- Alice si Bob decripteaza impreuna rezultatul.

Criptarea circuitului. Presupunem ca Alice vrea sa cripteze o poarta conjunctiva:

$\cdot$	0	1
0	0	0
1	0	1

Presupunem ca in aceasta poarta intra doua cabluri w_a si w_b si iese un cablu w_c. Alice inlocuieste valorile 0 si 1 pentru cablul w_a cu cuvinte binare random diferite X_a^0 si X_a^1. Ea procedeaza la fel pentru cablurile w_b si w_c. Tabela acestei porti arata acum asa:

$\cdot$	X_a^0	X_a^1
X_b^0	X_c^0	X_c^0
X_b^1	X_c^0	X_c^1

Aceasta tabela este acum criptata intr-un grup de 4 texte cifrate dupa cum urmeaza:

$$Enc_{X_a^0||X_b^0}(AX_c^0),$$
$$Enc_{X_a^1||X_b^0}(AX_c^0),$$
$$Enc_{X_a^0||X_b^1}(AX_c^0),$$
$$Enc_{X_a^1||X_b^1}(AX_c^1).$$

Aici A este o marca a descifrarii corecte, de exemplu textul *Valoarea corecta este* . Atunci cand alege criptarile aleatoare X_c^i, Alice trebuie sa se convinga ca numai:

$$Dec_{X_a^0||X_b^0}(Enc_{X_a^0||X_b^0}(AX_c^0)) = AW,$$

in timp ce celelalte chei $X_a^1||X_b^0$, $X_a^0||X_b^1$ si $X_a^1||X_b^1$ nu dau o decriptare de acest gen. Acest lucru este posibil cu mare probabilitate si se realizeaza intr-un numar mic de incercari, in general din prima incercare.

Aceste cuvinte sunt permutate aleator. Pozitia lor nu ofera nicio informatie cu privire la valori.

Transferul de date. Alice ii trimite lui Bob tabelele criptate. De asemeni, Alice ii trimite lui Bob criptarea inputului a. De exemplu, daca $a = a_4a_3a_2a_1a_0 = 11100$ atunci Alice transmite cuvintele binare $X_{a_4}^1$, $X_{a_3}^1$, $X_{a_2}^1$, $X_{a_1}^0$ si $X_{a_0}^0$. Bob are nevoie si de criptarea propriului sau input intr-un mod in care alice nu il afla. Folosim protocolul de transfer partial. Sa zicem ca inputul

lui Bob este $b = b_4b_3b_2b_1b_0 = 00101$. Bob intreaba initial $b_0 = 1$ intre stringurile stabilite de Alice $X^0_{b_0}$ si $X^1_{b_0}$. Prin transfer partial de tip *unul din doi*, el il obtine pe $X^1_{b_0}$ fara sa il poata decripta niciodata pe $X^0_{b_0}$, iar Alice nu afla niciodata de care dintre cele doua stringuri are nevoie Bob.

Evaluare. Acum Bob parcurge tabelele criptate si este capabil sa decripteze din fiecare tabela o singura linie, si anume:

$$X_c = Dec_{X_a,X_b}(Enc_{X_a,X_b}(AX_c)),$$

exact acea linie pentru care obtine un cuvat de forma AW. Bob continua aceasta evaluare pana cand obtine criptarile bitilor de output.

Sfarsit. Pentru fiecare bit de iesire x, Alice destainuie semnificatia lui X^0_x si X^1_x, iar Bob destainuie care dintre ele a fost calculat pe poarta respectiva de output. Astfel, ambii afla rezultatul. □

44 Circuite aritmetice

In capitolul precedent am prezentat criptarea circuitelor booleene in vederea calculului partajat bipartit. Aici vom cripta circuite aritmetice in vederea calculului criptat multipartit. Protocolul are la baza metoda Shamir's Secrete Sharing.

Sa presupunem ca n useri $A_1, \ldots, A_n$ vor sa calculeze impreuna o functie aritmetica $f(x_1, \ldots, x_n)$, unde fiecare x_i este valoarea adusa de A_i. Aceasta valoare trebuie sa ramana secreta pentru ceilalti participanti la calcul dar cu totii trebuie sa aiba incredere in rezultat, care este interesul lor comun. Asemenea functii se pot calcula prin circuite aritmetice. De exemplu, daca $n = 6$, functia:

$$f(x_1, \ldots, x_6) = x_1x_2 + x_3x_4 + x_5x_6$$

se calculeaza folosind un circuit cu 3 porti multiplicative si 2 porti aditive.

Cum in aplicatiile practice, domeniul de interes este limitat, se poate intotdeauna alege un numar prim p foarte mare, gen 2^{1024}, si se poate considera ca numeric toate $x_i = x_i \bmod p$, acelasi lucru se intampla si cu valorile intermediare ale calculului, iar in final $f(x_1, \ldots, x_n) = f(x_1, \ldots, x_n) \bmod p$. I acest caz, inlocuim portile aditive si multiplicative cu efectuarea adunarii si respectiv a inmultirii cu operatiile $+ \bmod p$ si $\times \bmod p$.

Partajarea valorilor. Fiecare participant A_i alege random $f_1, \ldots, f_t \in \mathbb{F}_p$ si construieste un polinom de partajare:

$$h_i(x) = x_i + f_1x + \cdots + f_tx^t.$$

Userul A_j primeste de la userul A_i o partitie a valorii sale secrete x_i, si anume:

$$x_i^{(j)} = h_i(j).$$

Asta se intampla si pentru $i = j$. Fiecare user evalueaza circuitul ca mai jos, folosind doar valorile pe care le-a primit, deci userul j va folosi valorile:

$$x_1^{(j)}, x_2^{(j)}, \ldots, x_n^{(j)}.$$

Portile aditive: La portile aditive, pur si simplu se aduna modulo n. Asta se vede in modul urmator. Sa spunem ca doua valori partajate $a^{(i)}$ si $b^{(i)}$ ajung la poarta aditiva, si ele au fost partitionate cu polinoame:

$$f(x) = a + f_1x + \cdots + f_tx^t,$$
$$g(x) = b + g_1x + \cdots + g_tx^t.$$

Daca $a + b = c \bmod p$ si $f + g = h$ in $\mathbb{Z}_p[x]$, observam ca:

$$c^{(i)} = h(i) = (f(i) + g(i)) = a^{(i)} + b^{(i)},$$

deci fiecare user trebuie sa adune valorile partajate si va obtine o valoare care este partajul respectiv al sumei.

Portile multiplicative: Intai de toate sa ne reamintim o proprietate a interpolarii Lagrange peste corpuri. Daca $f(x)$ este un polinom si partajam valorile $f(j)$, atunci exista un vector, $(r_1, \ldots, r_n)$, astfel incat:

$$f(0) = \sum_{i=1}^{n} r_i f(i),$$

si unul si acelasi vector functioneaza pentru toate polinoamele de grad $\leq n-1$. Numim acest vector, *vector de recombinare*.

Presupunem ca fiecare parte are o valoare partajata pentru a si b prin $a^{(i)} = f(i)$ si $b^{(i)} = g(i)$, where $f(0) = a$ and $g(0) = b$. Vrem sa calculam o valoare partajata $c^{(i)} = h(i)$ astfel incat pentru un anumit polinom $h(x)$ comun tuturor userilor, $h(0) = c = ab$. Iata cum se procedeaza:

1. Fiecare user calculeaza local $d^{(i)} = a^{(i)} b^{(i)}$.
2. Fiecare user produce un polinom $\delta_i(x)$ de grad cel mult t astfel incat $\delta_i(0) = d(i)$.
3. Fiecare user i distribuie fiecarui user j (inclusiv lui insusi) valoarea $d_{i,j} = \delta_i(j)$.
4. Fiecare user j calculeaza acum:
$$c^{(j)} = \sum_{i=1}^{n} r_i d_{i,j}.$$

La pasul (1), se calculeaza un polinom $h'(x)$ de grad $\leq 2t$, cu $d_i = h'(i)$ si $c = h'(0)$. Daca:

$$2t \leq n-1,$$

atunci:

$$c = \sum_{i=1}^{n} r_i d^{(i)}.$$

Acum sa consideram polinoamele $\delta_i(x)$ generate la pasul (2), si la polinomul:

$$h(x) = \sum_{i=1}^{n} r_i \delta_i(x).$$

Cum $\delta_i(x)$ sunt toate de grad $\leq t$, asa este si polinomul $h(x)$. De asemeni:

$$h(0) = \sum_{i=1}^{n} r_i \delta_i(0) = \sum_{i=1}^{n} r_i d^{(i)} = c.$$

Pe de alta parte, partajarea facuta la ultimul pas se bazeaza in mod implicit pe acest polinom:

$$h(j) = \sum_{i=1}^{n} r_i \delta_i(j) = \sum_{i=1}^{n} r_i d_{i,j} = c^{(j)}.$$

Deci, daca $t < n/2$, acest protocol duce la rezultate partiale, care se pot recombina in ultimul pas prin formula de interpolare. □

45 Criptare fara cheie

Curios, exista posibilitatea unei comunicari criptate fara cheie. Securitatea este asigurata de dificultatea logaritmului discret. Iata cum functioneaza acest protocol.

Initializare. Fie p un numar prim foarte mare, cunoscut public. Pentru comunicare, Alice alege la intamplare un $a \in \mathbb{Z}_{p-1}^{\times}$ si calculeaza $a' \in \mathbb{Z}_{p-1}^{\times}$ astfel incat $aa' = 1 \bmod p-1$. La randul lui Bob alege la intamplare un $b \in \mathbb{Z}_{p-1}^{\times}$ si calculeaza $b' \in \mathbb{Z}_{p-1}^{\times}$ astfel incat $bb' = 1 \bmod p-1$. Alice si Bob nu fac niciun schimb de informatie in priviinta alegerii facute.

Desfasurare. Alice calculeaza $c = m^a \bmod p$ si i-l trimite lui Bob. Bob calculeaza $d = c^b \bmod p$ si i-l trimite lui Alice. Alice calculeaza $e = d^{a'} \bmod p$ si i-l trimite lui Bob. Bob calculeaza $m = e^{b'} \bmod p$, si a decriptat mesajul.

Intr-adevar,

$$e^{b'} \bmod p = d^{a'b'} \bmod p = c^{ba'b'} \bmod p = m^{aba'b'} \bmod p = m^{k(p-1)+1} \bmod p = m \bmod p.$$

In cazul putin probabil, in care $d = m$, Bob va ii trimite lui Alice un cuvant aleator pentru a nu da de banuit ca a descifrat deja mesajul. Pentru a il ajuta sa observe aceasta situatie rara, mesajul initial trebuie completat cu o cheie de recunoastere, cunoscuta numai de Alice si Bob. □

Criptarea $c = m^a \bmod p$ cu descifrarea $m = c^{a'} \bmod p$ se numeste criptare Pohlig - Hellman.