Securitatea Informaţiei

Vol. 2

(Protocoale de securitate)

Adrian Atanasiu

Editura INFODATA Cluj

Prefaţă

Scrierea unei continuări la primul volum de securitate a informaţiei (dedicat sistemelor de criptare) se impunea de la sine: domeniul este mult mai vast decât o simplă trecere în revistă a principalelor sisteme care asigură confidenţialitatea şi autenticitatea mesajelor. Şi se extinde în mod exploziv, cuprinzând zone care până acum câţiva ani păreau complet neinteresate de o prelucrare electronică a informaţiei (mă gândesc la domeniile legislativ, medical, biologic; şi nu numai).

Au apărut în România destul de multe firme care produc soft de securitate. Există şi o legislaţie care reglementează o serie de criterii de performanţă ce trebuie îndeplinite pentru a ridica aceste produse la standard european. Realizatorii acestor softuri sunt informaticieni (absolvenţi ai facultăţilor de profil), precum şi matematicieni. Ca urmare, aproape toate facultăţile de informatică au introdus programe de studiu care se ocupă de securitatea informaţiei.
La Universitatea din Bucureşti există un curs obligatoriu în cadrul licenţei, dedicat Criptografiei, precum şi direcţii de master. Volumul actual constituie suportul unui astfel de curs, predat studenţilor din anul II, Masterul de Sisteme Distribuite şi Securitate.

Aş dori să fac câteva observaţii referitoare la acest volum:

- Toate noţiunile prezentate sunt la nivel teoretic.
  A fost o necesitate legată de omogenitatea lucrării. Este extrem de dificil de urmărit în paralel un protocol – împreună cu justificarea securităţii lui folosind noţiuni de complexitate, şi modul prin care poate fi el implementat, bazat pe utilizarea de anumite platforme (deci cu o portabilitate relativ restrânsă) destinate asigurării de diverse facilităţi. În cazul unor anumite protocoale însă, am prezentat şi unele standarde de implementare, dar tot sub o formă teoretică, ca să poată fi abordată o fundamentare a securităţii acestor implementări.

- Nu este o carte uşor de citit.
  Noţiunile folosite sunt complexe şi necesită un volum bogat de cunoştinţe de matematică şi informatică teoretică. Aproape toate protocoalele de securitate dezvoltate în acest moment se bazează pe problema logaritmului discret, cu diverse probleme conexe derivate din ea, cum ar fi problema Diffie - Hellman (în mai multe variante) sau problema reprezentării (baza construirii sistemului electronic de plată Brands

– implementat sub numele de DigiCash). Ca aparat matematic, în ultimul deceniu s-a dezvoltat calculul pe curbe eliptice, impreună cu un obiect extrem de sofisticat şi dificil (mai ales de înţeles) – cel de pereche biliniară (Tate şi Weil).

- Prezentul volum este dedicat protocoalelor de securitate bazate pe primitive teoretice. Din acest motiv, în forma sa iniţială el trebuia să mai conţină două capitole: unul de *Securitatea fluxului informaţional*, iar celălalt – de noţiuni legate de *Zero-knowledge*, inclusiv de *bit-commitment*. Lipsa acestui ultime noţiuni de exemplu a decimat capitolul legat de votul electronic (majoritatea acestor protocoale fiind bazate pe teste de tip bit-commitment). Diverse motive (depăşirea numărului de pagini, lipsa de timp, programa de un semestru a cursului) au făcut să renunţ la aceste capitole, care îşi vor găsi locul în unul din volumele următoare.
  Evident, există şi alte teme care ar putea fi încadrate în această zonă. Mă refer la modele Dolev - Yao, la tratarea securităţii informaţiei din perspectiva logicii temporale, elemente de control policy, sau modele de securitate cuantice şi biologice. Poate mai târziu ...

În forma sa actuală, acest volum este segmentat în 8 capitole. Primul este dedicat unei primtive criptografice extrem de utilă în construirea celorlaltor structuri: funcţiile de dispersie (hash).

Următoarele două capitole se referă la modalităţi de autentificare, atât sub o formă slabă (MAC-uri şi canale subliminale) cât şi sub o formă mai puternică – cea a semnăturilor electronice.

Capitolele 4 şi 5 pot constitui ele singure subiecte ale unor module de curs. Este vorba de scheme de partajare a secretelor şi de generare a cheilor de sesiune. A fost necesară o muncă deosebit de grea de selecţie şi clasificare a acestor protocoale, precum şi de alegere a celor mai reprezentative scheme pentru a fi prezentate.

Capitolul 6 este – într-un fel – o noutate. Prezintă o serie de algoritmi de criptare rapizi, bazaţi pe autentificarea partenerilor. Este o temă în plină dezvoltare, care foloseşte din plin avantajele criptografiei pe curbe eliptice.

Capitolele 7 şi 8 au o tentă ceva mai practică; ele se referă la protocoale de plată utilizate în comerţul electronic, respectiv protocoale de vot electronic. Sunt de asemenea teme în care mai sunt multe de spus. Astfel, nu am pomenit nimic despre protocoale $B2B$ (Bussiness to Bussiness) sau $B2C$ (Bussiness to Consumer), despre portofele electronice sau despre standardele utilizate în comerţul electronic. De asemenea, nu am scris despre algoritmii folosiţi în maşinile de vot – inclusiv de algoritmul Chaum (un algoritm superb de partajare a votului, bazat pe criptografia vizuală).
Deci şi aici mai sunt multe completări de făcut.

În volumele următoare, sper.

Prof. Dr. Adrian Atanasiu
15 august 2009

Cuprins

Capitolul 1

Funcţii de dispersie

1.1 Noţiuni preliminare

O funcţie de dispersie (funcţie *hash* în engleză) este – în esenţă o funcţie de compresie al cărei principal scop este asigurarea integrităţii unui mesaj. Cu ajutorul ei se construieşte o "amprentă" a mesajului, care – prin verificare periodică – certifică dacă acesta a fost modificat sau nu. De remarcat că nu se pune problema confidenţialităţii mesajului, ci doar a integrităţii şi autenticităţii sale.
Formal:

Definiţia 1.1. *O clasă de dispersie este un quadruplu* $(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$ *unde:*

- $\mathcal{X}$ *este mulţimea mesajelor posibile (criptate sau nu);*
- $\mathcal{Y}$ *este o mulţime finită de "amprente" (sau "rezumate");*
- $\mathcal{K}$ *este o mulţime finită de chei;*
- *Pentru fiecare cheie* $K \in \mathcal{K}$ *există o funcţie de dispersie* $h_K \in \mathcal{H}$, $h_K : \mathcal{X} \longrightarrow \mathcal{Y}$.

Ideea de "compresie" vine din faptul că – deşi $\mathcal{X}$ poate fi o mulţime oricât de mare – $\mathcal{Y}$ este o mulţime finită. Vom avea totdeauna $card(\mathcal{X}) \geq card(\mathcal{Y})$. Aproape toate aplicaţiile vor solicita o condiţie mai tare:

$$card(\mathcal{X}) \geq 2 \cdot card(\mathcal{Y}).$$

O pereche $(x, y) \in \mathcal{X} \times \mathcal{Y}$ este *validă pentru cheia* K dacă $h_K(x) = y$.

Observaţia 1.1. *Deoarece principalul scop urmărit este cel de integritate, compresia nu este o regulă generală aplicată tuturor funcţiilor de dispersie. Există funcţii de dispersie în care* $card(\mathcal{X}) = card(\mathcal{Y})$. *Din acest punct de vedere, orice sistem de criptare poate fi privit ca o funcţie de dispersie. Totuşi, în acest capitol vom trata numai funcţii de dispersie care verifică şi condiţia de compresie.*

Vom considera următoarele notaţii:

$$card(\mathcal{X}) = N, \quad card(\mathcal{Y}) = M, \quad \mathcal{F}^{\mathcal{X},\mathcal{Y}} = \{f \mid f : \mathcal{X} \longrightarrow \mathcal{Y}\}$$

Evident $card(\mathcal{F}^{\mathcal{X},\mathcal{Y}}) = M^N$.
Orice clasă de dispersie $\mathcal{F} \subseteq \mathcal{F}^{\mathcal{X},\mathcal{Y}}$ se numeşte (N, M) - clasă de dispersie.

În general, o clasă de dispersie este caracterizată complet de perechea $(\mathcal{K}, h_K)$.

Cazul cel mai simplu este când $card(\mathcal{K}) = 1$, deci există o singură cheie şi o singură funcţie de dispersie $h : \mathcal{X} \longrightarrow \mathcal{Y}$. Atunci se poate identifica clasa de dispersie cu funcţia de dispersie h.[1]

1.2 Securitatea funcţiilor de dispersie

1.2.1 Funcţii de dispersie criptografice

Vom considera situaţia unei singure funcţii de dipersie $h : \mathcal{X} \longrightarrow \mathcal{Y}$. Din punct de vedere al securităţii (criptografice), unica modalitate de a obţine o pereche validă $(x, y) \in \mathcal{X} \times \mathcal{Y}$ este de a alege x şi apoi de a calcula $y = h(x)$.
Acest lucru este implicat de condiţia ca următoarele 3 probleme să fie dificile.

- **NIP** (Non-inversabilă): Fiind dat $y \in \mathcal{Y}$ este dificil de aflat $x \in \mathcal{X}$ astfel ca $y = h(x)$.
- **CSP** (Coliziuni slabe): Fiind dată o pereche validă (x, y), este dificil de aflat $x_1 \neq x$ cu $h(x_1) = h(x)$.
- **CP** (Coliziuni): Este dificil de aflat două valori distincte $x, x_1 \in \mathcal{X}$ astfel ca $h(x) = h(x_1)$.

Prin "*dificil*" se înţelege că pentru rezolvarea problemei respective nu se cunosc decât algoritmi de complexitate mare (eventual problema este $\mathcal{NP}$ - completă).

De remarcat că problema **CP** nu conduce direct la o pereche validă. Dar, dacă (x, y) este o pereche validă şi (x, x_1) este o coliziune, atunci (x_1, y) este de asemenea o pereche validă.

Definiţia 1.2. *O funcţie de dispersie $h : \mathcal{X} \longrightarrow \mathcal{Y}$ pentru care problema* **CP** *este dificilă se numeşte "rezistentă la coliziuni"*

Deoarece $N > M$, orice funcţie de dispersie admite coliziuni. Totul este ca aceste coliziuni să nu poată fi găsite decât extrem de greu, neavând la îndemână decât algoritmi de complexitate mare.

Evident, dacă o funcţie este rezistentă la coliziuni, atunci ea va rezista şi la coliziuni slabe. Deci **CP** implică **CSP**.

[1] În literatură, acest caz se numeşte "unkeyed hash function".

Teorema 1.1. *Fie $h : \mathcal{X} \longrightarrow \mathcal{Y}$ o funcţie de dispersie, unde $\mathcal{X}, \mathcal{Y}$ sunt mulţimi finite, $N = card(\mathcal{X})$, $M = card(\mathcal{Y})$, $N \geq 2M$. Să presupunem că există un algoritm $\mathbf{A}$ de inversare a lui h. Atunci va exista un algoritm probabilist Las Vegas care găseşte o coliziune pentru h cu probabilitate cel puţin $\frac{1}{2}$.*

Demonstraţie. Fie $\mathbf{A}$ algoritmul de inversiune pentru h, de forma unui oracol $\mathbf{A}$ care admite la intrare o amprentă $y \in \mathcal{Y}$ şi întoarce un element $\mathbf{A}(y) \in X$ astfel ca $h(\mathbf{A}(y)) = y$.

Să considerăm algoritmul următor (notat cu $\mathbf{B}$):

1. Fie $x \in \mathcal{X}$ ales aleator;
2. $y \longleftarrow h(x)$;
3. $x_1 \longleftarrow \mathbf{A}(y)$
4. **if** $x_1 \neq x$ **then** x_1 şi x formează o coliziune pentru h (succes)
   else sfârşit (eşec)

$\mathbf{B}$ este un algoritm probabilist de tip *Las Vegas*, deoarece întoarce o coliziune sau nici un răspuns([2], pag. 147). Deci, este suficient să calculăm probabilitatea sa de succes.

Pentru orice $x, x_1 \in \mathcal{X}$ se defineşte $x \sim x_1$ dacă $h(x) = h(x_1)$.

Este uşor de demonstrat că $\sim$ este relaţie de echivalenţă. Definim

$$[x] = \{x_1 \mid x_1 \in \mathcal{X}, \ \ x \sim x_1\}$$

Fiecare clasă de echivalenţă $[x]$ este formată din mesaje care produc aceeaşi amprentă ca şi x. Numărul de clase de echivalenţă este deci cel mult M.
Fie $\mathcal{C} = \mathcal{X}/\sim$ mulţimea claselor de echivalenţă.

Presupunem că x este ales aleator din $\mathcal{X}$ la pasul 1. Pentru acest mesaj pot apare (la pasul 3) $n = card([x])$ valori x_1 posibile. $n - 1$ din ele sunt diferite de x şi duc la reuşita algoritmului în pasul 4.
Pentru o alegere particulară a lui x, probabilitatea de succes este deci $\dfrac{n-1}{n}$.

Probabilitatea de succes a algoritmului este calculată ca medie peste toate alegerile posibile ale lui x:

$$Pr_{succes} = \frac{1}{N}\sum_{x\in\mathcal{X}}\frac{n-1}{n} = \frac{1}{N}\sum_{c\in\mathcal{C}}\sum_{x\in c}\frac{card(c)-1}{card(c)} = \frac{1}{N}\sum_{c\in\mathcal{C}}(card(c)-1)$$
$$= \frac{1}{N}\left(\sum_{c\in\mathcal{C}} card(c) - \sum_{c\in\mathcal{C}} 1\right) \geq \frac{N-M}{N} \geq \frac{N-N/2}{N} = \frac{1}{2}$$

Probabilitatea de succes este deci cel puţin $\frac{1}{2}$. □

Din această teoremă se deduce:

Lema 1.1. CP *implică* **NIP**.

Rezultă că este suficient ca o funcţie de dispersie să fie cu coliziuni tari, aceasta implicând celelalte două proprietăţi de securitate.
De aceea, orice funcţie de dispersie trebuie să satisfacă problema **CP**.

Definiţia 1.3. *O funcţie de dispersie care verifică* **CP** *se numeşte "funcţie de dispersie criptografică".*

1.2.2 Atacul naşterilor

O altă condiţie de securitate – care impune o margine inferioară pentru $M = card(\mathcal{Y})$ – rezultă dintr-o metodă simplă de obţinere a coliziunilor, numită *atacul naşterilor*; numele provine de la *paradoxul naşterilor*. Sub o formă simplificată, acesta este:

Probabilitatea ca dintr-o mulţime aleatoare de 23 persoane să existe doi indivizi cu aceeaşi zi de naştere, este cel puţin 1/2.

Demonstraţie. Fie $\mathcal{X}, \mathcal{Y}$ două mulţimi finite, $card(\mathcal{X}) = N,\ card(\mathcal{Y}) = M; N \geq 2M$ şi fie $h : \mathcal{X} \longrightarrow \mathcal{Y}$ o funcţie de dispersie.
Este uşor de arătat că există cel puţin N coliziuni, dar problema este de a le pune în evidenţă.
O metodă simplă constă în a extrage aleator k mesaje distincte $x_1, \ldots, x_k \in \mathcal{X}$, de a calcula $y_i = h(x_i),\ 1 \leq i \leq k$ şi de a căuta dacă există o coliziune printre ele (făcând, de exemplu, o triere printre valorile y_i).

Cum x_i sunt extrase aleator, se poate considera că şi y_i sunt elemente aleatoare (nu neapărat distincte) din $\mathcal{Y}$. Probabilitatea ca din k extrageri, toate elementele y_i să fie distincte, se determină astfel:

Se face o ordonare a numerelelor y_i; fie $y_1, \ldots, y_k$ această ordine.
Prima extragere y_1 este total aleatoare; probabi-litatea ca $y_2 \neq y_1$ este $1 - 1/N$, probabilitatea ca y_3 să fie distinct de y_1 şi y_2 este $1 - 2/N$ etc.

Probabilitatea să nu avem k coliziuni va fi deci:

$$\left(1 - \frac{1}{N}\right)\left(1 - \frac{2}{N}\right)\ldots\left(1 - \frac{k-1}{N}\right) = \prod_{i=1}^{k-1}\left(1 - \frac{i}{N}\right)$$

Dacă x este un număr real suficient de mic, atunci se poate folosi aproximarea (rezultată din dezvoltarea în serie *Mac Laurin*) $1 - x \approx e^{-x}$. Vom obţine atunci:

$$\prod_{i=1}^{k-1}\left(1 - \frac{i}{N}\right) \approx \prod_{i=1}^{k-1} e^{\frac{-i}{N}} = e^{\frac{-k(k-1)}{2N}}.$$

Probabilitatea să existe cel puţin o coliziune este atunci $1 - e^{\frac{-k(k-1)}{2N}}$.
Notând această probabilitate cu ϵ, se obţine ecuaţia de necunoscută k : $e^{\frac{-k(k-1)}{2N}} \approx 1-\epsilon$, care conduce la $k^2 - k \approx 2N \cdot ln\frac{1}{1-\epsilon}$.

Dacă se ignoră termenul $-k$, se obţine $k \approx \sqrt{2N \cdot ln\frac{1}{1-\epsilon}}$.

Luând acum $\epsilon = 0.5$, se ajunge la $k \approx 1.17\sqrt{N}$.

De remarcat că alte valori pentru ϵ conduc la valori diferite pentru k, dar multiplicate totdeauna cu un factor constant $\sqrt{N}$.

Dacă $\mathcal{X}$ este mulţimea fiinţelor umane, Y este mulţimea celor 365 zile dintr-un an (nebisect) iar $h(x)$ reprezintă data de naştere a persoanei x, se obţine paradoxul naşterilor. În aproximarea de sus, cu $N = 365$ avem $k \approx 22.5$.
Deci, printre 23 persoane alese întâmplător, probabilitatea ca două să aibă aceeaşi zi de naştere este $1 - 1/2$. □

Atacul naşterilor este un atac prin forţă brută care urmăreşte aflarea unor coliziuni. În general, *Oscar* generează aleator mesajele $x_1, x_2, \ldots \in \mathcal{X}$. Pentru fiecare mesaj x_i el calculează şi stochează amprenta $y_i = h(x_i)$, comparând-o cu valorile stocate anterior.[2] Dacă $h(x_i)$ coincide cu o valoare $h(x_j)$ deja stocată, *Oscar* a găsit o coliziune (x_i, x_j). Conform paradoxului naşterilor, aceasta se va întâmpla după aproximativ $2^{N/2}$ mesaje.

Acest tip de atac impune – ca măsură de securitate – o margine inferioară asupra lungimii amprentelor.
O amprentă de 40 biţi este vulnerabilă la un atac al naşterilor folosind numai 2^{20} (aproximativ un milion) mesaje aleatoare.
Se sugerează de aceea folosirea de amprente de cel puţin 256 biţi (aici atacul naşterilor va cere 2^{128} calcule).

Exemplul 1.1. *Unele sisteme de semnătură digitală (care vor fi studiate în Capitolul 3) se aplică unei amprente $h(x)$ a mesajului x. Să presupunem că Oscar şi Bob semnează un contract.*

1. *Oscar pregăteşte două versiuni ale contractului: o versiune m_1 echitabilă, şi o versiune m_2 avantajoasă lui Oscar – ambele de dimensiune n.*

2. *Apoi generează $\mathcal{O}(2^{n/2})$ variante minore ale celor două versiuni, până obţine două mesaje m_1' (similar lui m_1) şi m_2' (similar cu m_2) având $h(m_1') = h(m_2')$.*
   De exemplu, dacă m_1 conţine o figură (bitmap sau jpeg), modificările sunt insesizabile.

3. *Cei doi semnează $h(m_1')$.*

Ulterior Oscar poate pretinde că semnătura se referă la contractul m_2'.

[2] În practică, mesajele sunt generate folosind un generator de numere pseudo-aleatoare. Deci mesajele pot fi recalculate oricând, nefiind necesară decât stocarea amprentelor.

1.2.3 Aplicaţii ale funcţiilor de dispersie

Funcţiile de dispersie sunt utilizate într-o paletă largă de aplicaţii criptografice. Listăm o mică parte din ele:

- *Detectare de viruşi*:

  Scopul este detectarea modificării unui fişier.
  - Pentru un fişier F se calculează $c = h(F)$, unde h este o funcţie de dispersie criptografică (este suficient să satisfacă **CSP**).
  - Valoarea c este trimisă printr-un canal sigur (de exemplu o dischetă sau un CD) utilizatorului care suspectează coruperea fişierului primit.
  - Acesta compară c cu $h(F')$, unde F' este fişierul primit.

- *Distribuţia de Software/Chei Publice*:

  Scopul este descărcarea de către utilizator a copiei "adevărate" de soft sau cheie publică.
  Ideea este similară cu cea anterioară.
  Utilizatorul obţine – via un canal securizat – amprenta produsului soft (sau a cheii publice) şi o compară cu amprenta produsă de el pentru acelaşi produs descărcat de pe Internet.

- *Stocări în baze de date nesigure*

  - Utilizatorul construieşte o tabelă cu amprente ale paginilor/documentelor, pe care le stochează într-o bază de date nesecurizată.
  - Apoi, când utilizează aceste pagini/documente, recalculează şi verifică amprentele lor la fiecare descărcare pe calculatorul propriu.

- *Securizarea referinţelor şi pointerilor*:

  - Orice referinţă la o pagină web este amprentată şi verificată atunci când se face apel la adresa respectivă.
  - Acelaşi procedeu se aplică şi URL-urilor.

- *Datare*:

  Toate fişierele sau documentele sunt amprentate, după care li se asociază o "ştampilă de timp" de către o unitate autorizată[3].

- *Parole One-time(S - Chei)*:

  - Plecând de la o valoare iniţială x_0, se generează o secvenţă de amprente:

$$x_0 \xrightarrow{h(x_0)} x_1 \xrightarrow{h(x_1)} x_2 \cdots \xrightarrow{h(x_{n-1})} x_n$$

[3]Detalii despre *Time Stamping* sunt prezentate în Capitolul 3.

- Sistemul original stochează x_n, iar utilizatorul foloseşte x_{n-1} ca parolă.
- Sistemul verifică $h(x_{n-1}) \stackrel{?}{=} x_n$. *Oscar* nu poate afla x_{n-1} chiar dacă ştie x_n, deoarece h este o funcţie de dispersie criptografică.
- În continuare, sistemul stochează x_{n-1}, iar utilizatorul foloseşte ca parolă x_{n-2}.

- *Generatori de numere pseudo-aleatoare*:

  Unele funcţii de dispersie asigură amprentelor o componentă aleatoare semnificativă, ceea ce le face utile – în anumite cazuri – pentru generarea de secvenţe de numere pseudo-aleatoare.

- *Coduri de autentificare a mesajelor (MAC)*:

  Vor fi prezentaţi detaliat în Capitolul 2.

1.3 Funcţia de dispersie Chaum-van Heijst-Pfitzmann

Funcţia de dispersie prezentată în această secţiune a fost definită în [16]. Ea nu este suficient de rapidă pentru aplicaţii practice dar constituie un exemplu foarte simplu de funcţie de dispersie care admite o probă de securitate pe o ipoteză rezonabilă.

Fie p un număr prim mare astfel ca $q = (p-1)/2$ să fie de asemenea prim.

Fie $\alpha, \beta \in Z_p$ două elemente primitive.

Valoarea $log_\alpha \beta$ nu este publică şi se presupune că este dificil de obţinut.

Funcţia de dispersie *Chaum-van Heijst-Pfitzmann* $h : Z_q \times Z_q \longleftarrow Z_p^*$ este definită prin:

$$h(x_1, x_2) = \alpha^{x_1} \beta^{x_2} \ (mod\ p)$$

Teorema 1.2. *Fiind dată o coliziune pentru funcţia de dispersie Chaum-van Heijst-Pfitzmann, calculul lui $log_\alpha \beta$ este uşor.*

Demonstraţie. Să presupunem că a apărut o coliziune

$$h(x_1, x_2) = h(x_3, x_4) \text{ cu } (x_1, x_2) \neq (x_3, x_4)$$

Avem deci $\alpha^{x_1} \beta^{x_2} \equiv \alpha^{x_3} \beta^{x_4} \ (mod\ p)$, sau $\alpha^{x_1 - x_3} \equiv \beta^{x_4 - x_2} \ (mod\ p)$.

Vom nota $d = (x_4 - x_2, p-1)$.

Cum $p - 1 = 2q$ cu q număr prim, rezultă $d \in \{1, 2, q, p-1\}$.

Să trecem în revistă fiecare din aceste patru posibilităţi.

- $\mathbf{d = 1}$: Notând $y = (x_4 - x_2)^{-1} \ (mod\ p-1)$, vom avea

  $$\beta \equiv \beta^{(x_4 - x_2)y} \ (mod\ p) \equiv \alpha^{(x_1 - x_3)y} \ (mod\ p)$$

  şi se poate calcula logaritmul discret $log_\alpha \beta = (x_1 - x_3)(x_4 - x_2)^{-1} \ (mod\ p-1)$.

- $\mathbf{d = 2}$: Deoarece $p - 1 = 2q,\ q$ prim, deducem $(x_4 - x_2, q) = 1$.

  Dacă notăm $y = (x_4 - x_2)^{-1}\ (mod\ q)$, atunci $(x_4 - x_2)y = kq + 1$ pentru un număr întreg k.
  Calculând, se obţine

  $$\beta^{(x_4-x_2)y} \equiv \beta^{kq+1}\ (mod\ p) \equiv (-1)^k\beta\ (mod\ p) \equiv \pm\beta\ (mod\ p)$$

  deoarece $\beta^q \equiv -1\ (mod\ p)$.

  Deci, $\alpha^{(x_1-x_3)y} \equiv \beta^{(x_4-x_2)y}\ (mod\ p) \equiv \pm\beta\ (mod\ p)$.

  De aici se deduce $log_\alpha\beta = (x_1 - x_3)(x_4 - x_2)^{-1}\ (mod\ p - 1)$, sau

  $$log_\alpha\beta = (x_1 - x_3)(x_4 - x_2)^{-1} + q\ (mod\ p - 1).$$

  Se poate verifica uşor care din aceste două rezultate este cel corect.

- $\mathbf{d = q}$: Din $0 \le x_2 \le q - 1,\ 0 \le x_4 \le q - 1$ rezultă $-(q - 1) \le x_4 - x_2 \le q - 1$.
  Este deci imposibil să avem $(x_4 - x_2, p - 1) = q$.

- $\mathbf{d = p - 1}$: Această variantă este posibilă numai dacă $x_4 = x_2$. Avem atunci $\alpha^{x_1}\beta^{x_2} \equiv \alpha^{x_3}\beta^{x_2}\ (mod\ p),\ $ deci $\alpha^{x_1} \equiv \alpha^{x_3}\ (mod\ p)$, de unde rezultă $x_1 = x_3$.
  S-a ajuns la $(x_1, x_2) = (x_3, x_4)$, ceea ce contrazice ipoteza.

În concluzie, h este cu coliziuni tari, depinzând de calculul lui $log_\alpha\beta$ în Z_p. □

Exemplul 1.2. *Să luăm* $p = 12347$, *deci* $q = 6173,\ \alpha = 2,\ \beta = 8461$.
Presupunem că s-a găsit coliziunea

$$\alpha^{5692}\beta^{144} \equiv \alpha^{212}\beta^{4214}\ (mod\ 12347).$$

Deci $x_1 = 5692,\ x_2 = 144,\ x_3 = 212,\ x_4 = 4214$. *Avem* $(x_4 - x_2, p - 1) = 2$
şi se începe calculul:

$$y = (x_4 - x_2)^{-1}\ (mod\ q) = (4214 - 144)^{-1}\ (mod\ 6173) = 4312$$

Calculăm apoi

$$y' = (x_1 - x_3)y\ (mod\ p - 1) = (5692 - 212)4312\ (mod\ 12346) = 11862.$$

Se obţine $log_\alpha\beta \in \{y', y' + q\ (mod\ p - 1)\}$.
Într-adevăr,

$$\alpha^{y'}\ (mod\ p) = 2^{11862}\ (mod\ 12346) = 9998,$$

deci

$$log_\alpha\beta = y' + q\ (mod\ p - 1) = 11862 + 6173\ (mod\ 12346) = 5689.$$

Se verifică imediat că $2^{5689} \equiv 8461\ (mod\ 12347)$.

1.4 Construcţia Merkle - Damgard a unei funcţii de dispersie

Vom prezenta o modalitate de construcţie a unei funcţii de dispersie, bazată pe compresie. În particular, ea permite extensia funcţiei de dispersie pe un domeniu infinit, păstrând proprietăţile de securitate; avantajul unei asemenea situaţii este acela că se pot amprenta mesaje de lungime arbitrară.

Construcţia Merkle - Damgard se referă la mesaje peste un alfabet binar $Z_2 = \{0, 1\}$.

Fie $h : Z_2^m \longrightarrow Z_2^t$ o funcţie criptografică de dispersie, cu $m \geq t + 1$. Ea face de fapt o compresie a blocurilor de m biţi în blocuri de t biţi.
Scopul construcţiei este realizarea unei funcţii de dispersie $h^* : \mathcal{X} \longrightarrow Z_2^t$, unde

$$\mathcal{X} = \bigcup_{i=m}^{\infty} Z_2^i$$

Să considerăm la început cazul $m \geq t + 2$.

Fiecare element al lui $\mathcal{X}$ este reprezentat printr-un şir de biţi; vom nota lungimea şirului x cu $|x|$, iar concatenarea a două şiruri $x, y \in \mathcal{X}$ cu $x\|y$.

Fie $|x| = n \geq m$; atunci x se poate scrie sub forma:

$$x = x_1\|x_2\| \ldots \|x_k$$

unde $\quad |x_1| = |x_2| = \ldots = |x_{k-1}| = m - t - 1, \quad |x_k| = m - t - 1 - d,\ 0 \leq d \leq m - t - 2.$
În acest fel se obţine $k = \left\lceil \dfrac{n}{m - t - 1} \right\rceil$. Definim funcţia $h^*(x)$ cu algoritmul următor:

1. **for** $i := 1$ **to** $k - 1$ **do** $y_i \longleftarrow x_i$
2. $y_k \longleftarrow x_k\|0^d$
3. $y_{k+1} \longleftarrow [d]_2$ (reprezentarea binară a lui d pe $m - t - 1$ biţi).
4. $g_1 \longleftarrow h(0^{t+1}\|y_1)$
5. **for** $i = 1$ **to** k **do** $g_{i+1} \longleftarrow h(g_i\|1\|y_{i+1})$
6. $h^*(x) \longleftarrow g_{k+1}$

Notăm

$$y(x) = y_1\|y_2\| \ldots \|y_{k+1}$$

y_k se obţine completând x_k la dreapta cu d zerouri, astfel ca toate blocurile y_i $(1 \leq i \leq k)$ să fie de lungime $m - t - 1$.

De asemenea, la pasul 3, y_{k+1} este completat la stânga cu zerouri pentru a obţine lungimea $m-t-1$.

Pentru dispersia lui x se construieşte $y(x)$ apoi se tratează iterativ blocurile $y_1, y_2, \ldots, y_{k+1}$. Din construcţia lui y_{k+1} se asigură injectivitatea aplicaţiei y.

Teorema 1.3. *Fie $h : Z_2^m \longrightarrow Z_2^t$ $(m \geq t+2)$ o funcţie de dispersie criptografică. Funcţia $h^* : \bigcup_{i=m}^{\infty} Z_2^i \longrightarrow Z_2^t$ definită prin algoritmul de mai sus, este de asemenea o funcţie criptografică.*

Demonstraţie. Trebuie demonstrat că funcţia h^* verifică problema **CP**.
Să presupunem prin absurd că există $x \neq x'$ astfel ca $h^*(x) = h^*(x')$. Vom arăta cum, plecând de la această coliziune se poate construi în timp polinomial o coliziune pentru h, lucru imposibil deoarece h este o funcţie criptografică – deci cu coliziuni tari.

Fie $y(x) = y_1\|y_2\|\ldots\|y_{k+1}$ şi $y(x') = y'_1\|y'_2\|\ldots\|y'_{s+1}$, unde x şi x' sunt completate la pasul 2 cu d respectiv d' zerouri. S-a notat cu $g_1, \ldots, g_{k+1}$ respectiv $g'_1, \ldots, g'_{s+1}$ valorile calculate de paşii 4 şi 5.

Apar două cazuri:

1. $|x| \not\equiv |x'| \ (mod\ m-t-1)$.

   Atunci $d \neq d',\ y_{k+1} \neq y'_{s+1}$. Avem:

   $$h(g_k\|1\|y_{k+1}) = g_{k+1} = h^*(x) = h^*(x') = g'_{s+1} = h(g'_s\|1\|y'_{s+1}),$$

   care este o coliziune pentru h deoarece $y_{k+1} \neq y'_{s+1}$.

2. $|x| \equiv |x'| \ (mod\ m-t-1)$.

   Aici trebuie studiate două subcazuri:

   (2a). $|x| = |x'|$. Atunci $k = s,\ y_{k+1} = y'_{s+1}$. Similar primului caz,

   $$h(g_k\|1\|y_{k+1}) = g_{k+1} = h^*(x) = h^*(x') = g'_{k+1} = h(g'_k\|1\|y'_{k+1})$$

   Dacă $g_k \neq g'_k$, s-a aflat o coliziune pentru h.
   Să presupunem deci că $g_k = g'_k$. Vom avea:

   $$h(g_{k-1}\|1\|y_k) = g_k = g'_k = h(g'_{k-1}\|1\|y'_k)$$

   Fie s-a găsit o coliziune pentru h, fie $g_{k-1} = g'_{k-1},\ y_k = y'_k$.
   Dacă nu a fost coliziune, se poate continua până la

   $$h(0^{t+1}\|y_1) = g_1 = g'_1 = h(0^{t+1}\|y'_1)$$

   Cum $y_1 \neq y'_1$ înseamnă coliziune pentru h, să presupunem $y_1 = y'_1$. Vom avea $y_i = y'_i$ $(1 \leq i \leq k+1)$ deci $y(x) = y(x')$.

Cum aplicaţia y este injectivă, rezultă $x = x'$, ceea ce contrazice ipoteza.

(2b). $|x| \neq |x'|$.

Se poate presupune, fără a micşora generalitatea, că $|x| < |x'|$, deci $s > k$.
Vom proceda similar cu **(2a)**.
Dacă nu s-au găsit coliziuni pentru h, se va obţine şirul de egalităţi

$$h(0^{t+1}\|y_1) = g_1 = g'_{s-k+1} = h(g'_{s-k}\|1\|y'_{s-k+1})$$

Primii $t+1$ biţi din $0^{t+1}\|y_1$ conţin numai zerouri, în timp ce primii $t+1$ biţi din $g'_{s-k}\|1\|y'_{s-k+1}$ conţin un 1. Deci s-a găsit o coliziune pentru h.

Rezultă că pentru toate cazurile studiate, s-a ajuns la coliziuni.

□

Algoritmul pe care s-a construit toată demonstraţia de sus este adevărat numai dacă $m \geq t+2$. Rămâne cazul $m = t+1$; aici se va realiza o construcţie diferită pentru h^*.

Ca anterior, fie $|x| = n \geq m$.

Vom începe prin a codifica x folosind funcţia f definită

$$f(0) = 0, f(1) = 01$$

Construcţia lui h^* se face pe baza următorului algoritm:

1. $y \longleftarrow y_1\|y_2\|\ldots\|y_k = 11\|f(x_1)\|f(x_2)\|\ldots\|f(x_n)$ $(y_i \in Z_2)$
2. $g_1 \longleftarrow h(0^t\|y_1)$
3. d **for** $i = 1$ **to** $k-1$ **do** $g_{i+1} \longleftarrow h(g_i\|y_{i+1})$
4. $h^*(x) \longleftarrow g_k$

Codificarea $x \mapsto y = y(x)$ din pasul 1 verifică câteva proprietăţi importante:

- y este injectivă
- y este liberă de prefix: nu există x, x', z astfel ca $y(x) = z\|y(x')$

Teorema 1.4. *Fie $h : Z_2^{t+1} \longrightarrow Z_2^t$ o funcţie de dispersie criptografică.*
Funcţia $h^* : \bigcup_{i=t+1}^{\infty} Z_2^i \longrightarrow Z_2^t$ este de asemenea o funcţie criptografică.

Demonstraţie. Să presupunem prin absurd că există $x \neq x'$ cu $h^*(x) = h^*(x')$.
Vom avea

$$y(x) = y_1 y_2 \dots y_k, \quad y(x') = y'_1 y'_2 \dots y'_s.$$

Apar două cazuri:

1. $k = s$.
   Similar cu demonstraţia teoremei anterioare, se găseşte fie o coliziune pentru h, fie $y(x) = y(x')$, ceea ce duce la $x = x'$, contradicţie.

2. $k \neq s$. Vom considera situaţia $s > k$. Modul de abordare a acestui caz este similar cu cele de mai sus. Presupunând că nu s-au găsit coliziuni pentru h, se obţine şirul de identităţi
   $$y_k = y'_s,\ y_{k-1} = y'_{s-1},\ \dots y_1 = y'_{s-k+1},$$
   ceea ce contrazice faptul că codificarea este liberă de prefix.

Deci h este cu coliziuni tari. □

Să rezumăm într-o singură teoremă rezultatele obţinute:

Teorema 1.5. *Fie $h : Z_2^m \longrightarrow Z_2^t$, $(m \geq t+1)$ o funcţie de dispersie criptografică. Există atunci o funcţie de dispersie criptografică*

$$h^* : \bigcup_{i=m}^{\infty} Z_2^i \longrightarrow Z_2^t.$$

Numărul de calcule al lui h pentru efectuarea unei aplicări a lui h^* este cel mult

$$\begin{cases} 1 + \left\lceil \dfrac{n}{m-t-1} \right\rceil & \text{dacă } m \geq t+2, \\ 2n+2 & \text{dacă } m = t+1, \end{cases}$$

unde $n = |x|$.

1.5 Funcţii de dispersie bazate pe sisteme de criptare

Deoarece multe metode de construcţie ale funcţiilor de dispersie (inclusiv cele de sus) sunt prea lente pentru o utilizare practică, sunt propuse şi alte variante. Una din ele constă în folosirea unui sistem de criptare pentru a genera o funcţie de dispersie criptografică.

Să presupunem că $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ este un sistem de criptare simetric, cu $\mathcal{P} = \mathcal{C} = \mathcal{K} = Z_2^n$. Pentru a evita atacul naşterilor se va lua $n \geq 128$ (deci nu se poate folosi DES).

Fie şirul de biţi

$$x = x_1 \| x_2 \| \dots \| x_k, \qquad x_i \in Z_2^n \quad 1 \leq i \leq k.$$

Dacă numărul de biţi nu este multiplu de n, x se va completa ca în paragraful anterior; pentru simplificare, nu vom trata aici această chestiune.

Ideea construcţiei constă în fixarea unei *valori iniţiale* g_0 şi calcularea iterativă a lui $g_1, g_2, \ldots, g_k$ astfel ca $g_i = f(x_i, g_{i-1})$, unde f este o funcţie care utilizează regulile de criptare. Rezultatul final al dispersiei este amprenta $h(x) = g_k$.

Au fost propuse mai multe funcţii de dispersie, unele din ele fiind sparte (independent de sistemul de criptare utilizat).
O securitate mai mare a fost probată pentru următoarele variante:

$$\begin{array}{ll} g_i = e_{g_{i-1}}(x_i) \oplus x_i & \text{(Mathyas - Meyer - Oseas)} \\ g_i = e_{g_{i-1}}(x_i) \oplus g_{i-1} & \text{(Davies - Meyer)} \\ g_i = e_{g_{i-1}}(x_i) \oplus x_i \oplus g_{i-1} & \text{(Miyaguchi - Preneel)} \\ g_i = e_{g_{i-1}}(x_i \oplus g_{i-1}) \oplus x_i & \\ g_i = e_{g_{i-1}}(x_i \oplus g_{i-1}) \oplus x_i \oplus g_{i-1} & \end{array}$$

Pentru un sistem de criptare mai general, cu $\mathcal{P} = \mathcal{C} = 2^n$ ($n \geq 128$), $\mathcal{K} = 2^k$ sunt cunoscute alte două modalităţi de construcţie de funcţii de dispersie (prezentate în [25] drept funcţii de compresie), implementate frecvent sub o formă combinată. Anume:

- *Funcţii de dispersie MDC (Modification Detection Code)*:
  Textul clar se reîmparte în blocuri de lungime $n + r$. Funcţia de dispersie

  $$h : Z_2^{n+r} \longrightarrow Z_2^n$$

  este definită astfel: Dacă $z = x \| y$, $|x| = n$, $|y| = k$, atunci $h(z) = e_y(x)$.

- *Funcţia Matyias - Meyer - Oseas* (inclusă în standardul ISO/IEC $10118 - 2$):

  1. Blocul $z \in \mathcal{X}$ de lungime $2n$ este spart în două jumătăţi: $z = x \| y$.
  2. Se determină cheia K folosind o funcţie $g : Z_2^n \longrightarrow Z_2^k$
     (uzual $K = g(y)$ se obţine din y cu ajutorul unei funcţii booleene).
  3. Amprenta lui z este $h(z) = e_K(x) \oplus x$.

Securitatea unor astfel de funcţii de dispersie este în strânsă dependenţă cu securitatea oferită de sistemul de criptare folosit.

1.6 Funcţii de dispersie bazate pe aritmetica modulară

Plecând de la ideea că există echipamente soft şi chiar hard deosebit de performante pentru realizarea de operaţii în aritmetică modulară, au fost propuse diverse funcţii de

dispersie bazate pe astfel de calcule. Singurul lor dezavantaj este viteza relativ mică de obţinere a amprentelor.

Cele mai cunoscute sunt cele din clasa $MASH$ (Modular Arithmetic Secure Hash). $MASH1$ a fost construit pentru a fi inclus în standardul ISO/IEC. El foloseşte un modul n de tip RSA (deci lungimea sa este esenţială pentru asigurarea securităţii). Mai mult, mărimea acestui modul (1024 biţi conform cerinţelor actuale) determină mărimile blocurilor care sunt prelucrate iterativ, precum şi mărimea amprentei.

Fie deci p, q două numere prime mari şi $n = pq$ un număr scris în binar pe M biţi. Definim mărimea N (numărul de biţi) a amprentei prin

$$N = 16 \cdot \left\lceil \frac{M}{16} \right\rceil$$

Mesajul de intrare este $x \in Z_2^b \quad (0 \leq b \leq 2^{N/2})$. Funcţia de dispersie $MASH1$ este construită astfel:

1. Se definesc $H_0 = \mathbf{0}$ şi $A = (11110000 \ldots 11110000)$, ambele scrise pe N biţi.
2. Se extinde x cu zerouri până la cel mai apropiat multiplu de $N/2$ şi se scrie $x = x_1 \ldots x_t, \ \ |x_i| = N/2$.
3. $x_{t+1} \longleftarrow b$, unde b este reprezentat pe $N/2$ biţi.
4. Pentru $i = 1, 2, \ldots, t$, blocul x_i se extinde la n biţi astfel: x_i se împarte în subblocuri de câte 4 biţi, după care se inserează biţii 1111 la începutul fiecărui subbloc.
   Pentru x_{t+1} se procedează analog, singura diferenţă fiind biţii inseraţi: 1010 în loc de 1111.
   Fie y_i extensia lui x_i $(1 \leq i \leq t+1)$.
5. **for** $i = 1$ **to** $t+1$ **do**
   $$H_i \longleftarrow ((((H_{i-1} \oplus y_i) \vee A)^2 \ (mod\ n)) \gg N) \oplus H_{i-1}$$
   unde $\gg$ reprezintă deplasarea ciclică spre dreapta (cu N biţi).
6. $h(x) \longleftarrow H_{t+1}$

$MASH2$ are ca unică diferenţă înlocuirea exponentului $e = 2$ de la pasul 4. cu exponentul $e = 2^8 + 1$.

1.7 Funcţia de dispersie MD4

Funcţiile **MDi** (Message Digest nr. i) au fost construite de Ron Rivest. În particular, **MD4** a fost introdusă în 1990 ([68]) pentru procesoare pe 32 biţi. Deşi nu s-a dovedit suficient de sigură, principiile sale de construcţie au fost ulterior utilizate pentru o clasă largă de funcţii de dispersie, cum ar fi **MD5**, **SHA**, **SHA1**, **RIPEMD160**. Cu toate că securitatea lor nu a fost demonstrată (şi atacurile apărute de-a lungul timpului au reliefat diverse slăbiciuni de construcţie), funcţiile de dispersie **MDi** au avantajul de a fi foarte rapide, deci practice pentru semnarea de mesaje lungi.

1.7.1 Descrierea funcţiei MD4

Fiind dat un şir $x \in Z_2^*$, se defineşte tabloul $M = M_0 M_1 \dots M_{n-1}$ cu fiecare M_i (numit *cuvânt*) de lungime 32 şi $n \equiv 0 \ (mod\ 16)$.

M este construit folosind algoritmul următor:

1. $d \longleftarrow (447 - |x|) \ (mod\ 512)$
2. $s \longleftarrow$ reprezentarea binară a lui $|x| \ (mod\ 2^{64})$, $|s| = 64$
3. $M = x \| 1 \| 0^d \| s$

MD4 construieşte o amprentă a lui x pe 128 biţi. Algoritmul de generare este:

1. $A \longleftarrow (67452301_{16})$, $B \longleftarrow (efcdab89)_{16}$,
   $C \longleftarrow (98badcfe)_{16}$, $D \longleftarrow (10325476)_{16}$.
2. **for** $i = 0$ **to** $n/16 - 1$ **do**
   - 2.1. **for** $j = 0$ **to** 15 **do** $X_j \longleftarrow M_{16i+j}$
     - 2.1.1. $AA \longleftarrow A \quad BB \longleftarrow B \quad CC \longleftarrow C \quad DD \longleftarrow D$
     - 2.1.2. **Etapa 1**
     - 2.1.3. **Etapa 2**
     - 2.1.4. **Etapa 3**
     - 2.1.5. $A \longleftarrow A + AA \quad B \longleftarrow B + BB$
       $C \longleftarrow C + CC \quad D \longleftarrow D + DD$

Amprenta $h(x)$ este – în final – concatenarea celor patru cuvinte A, B, C, D (numite *registre*). Algoritmul împarte tabelul M în şiruri de câte 16 cuvinte consecutive, cărora le aplică trei etape de transformări. Adunările de la pasul 2.1.5 sunt efectuate modulo 2^{32}.

Fie $\alpha, \beta \in Z_2^*$, $|\alpha| = |\beta|$. Cele 3 etape se bazează pe următoarele operaţii:

- $\alpha \wedge \beta$ *şi* logic bit cu bit (AND)

- $\alpha \vee \beta$ *sau* logic bit cu bit (OR)
- $\alpha \oplus \beta$ *sau exclusiv* bit cu bit (XOR)
- $\neg\alpha$ complementară bit cu bit (NOT)
- $\alpha + \beta$ suma modulo 2^{32}
- $\alpha \ll s$ deplasare ciclică cu s biţi spre stânga ($1 \leq s \leq 31$)

Implementarea acestor operaţii se realizează direct pe hard şi ţine cont de arhitectura calculatorului. Să detaliem puţin acest lucru.

Fie $a_1a_2a_3a_4$ un cuvânt de patru octeţi, unde a_i este un număr întreg din intervalul $[0, 255]$.

- Într-o arhitectură *big-endian* (o staţie $SPARK$ de exemplu), un cuvânt reprezintă întregul $a_1 2^{24} + a_2 2^{16} + a_3 2^8 + a_4$
- Într-o arhitectură *little-endian* (cum este familia $Intel\ 80xxx$), un cuvânt reprezintă întregul $a_4 2^{24} + a_3 2^{16} + a_2 2^8 + a_1$

Construcţia standard a lui **MD4** s-a bazat pe o arhitectură *little-endian*. Cum amprenta rezultată în urma procesului de dispersie trebuie să fie independentă de arhitectura calculatorului folosit, la o implementare a lui **MD4** pe un calculator *big-endian*, adunările se vor defini astfel:

1. Se inter-schimbă $x_1 \leftrightarrow x_4,\ x_2 \leftrightarrow x_3,\ y_1 \leftrightarrow y_4,\ y_2 \leftrightarrow y_3$
2. Se calculează $Z = X + Y\ (mod\ 2^{32})$
3. Se inter-schimbă $z_1 \leftrightarrow z_4,\ z_2 \leftrightarrow z_3$

Cele trei etape de construcţie ale funcţiei de dispersie **MD4** folosesc trei funcţii $f, g, h : Z_2^{32} \times Z_2^{32} \times Z_2^{32} \longrightarrow Z_2^{32}$ definite:

$$f(X, Y, Z) = (X \wedge Y) \vee ((\neg X) \wedge Z)$$
$$g(X, Y, Z) = (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$$
$$s(X, Y, Z) = X \oplus Y \oplus Z$$

Descrierile celor trei etape sunt:

Etapa 1:

1. $A \longleftarrow (A + f(B, C, D) + X_0) \ll 3$
2. $D \longleftarrow (D + f(A, B, C) + X_1) \ll 7$
3. $C \longleftarrow (C + f(D, A, B) + X_2) \ll 11$
4. $B \longleftarrow (B + f(C, D, A) + X_3) \ll 19$
5. $A \longleftarrow (A + f(B, C, D) + X_4) \ll 3$
6. $D \longleftarrow (D + f(A, B, C) + X_5) \ll 7$
9. $A \longleftarrow (A + f(B, C, D) + X_8) \ll 3$
10. $D \longleftarrow (D + f(A, B, C) + X_9) \ll 7$
11. $C \longleftarrow (C + f(D, A, B) + X_{10}) \ll 11$
12. $B \longleftarrow (B + f(C, D, A) + X_{11}) \ll 19$
13. $A \longleftarrow (A + f(B, C, D) + X_{12}) \ll 3$
14. $D \longleftarrow (D + f(A, B, C) + X_{13}) \ll 7$

7. $C \longleftarrow (C + f(D,A,B) + X_6) \lll 11$ **15.** $C \longleftarrow (C + f(D,A,B) + X_{14}) \lll 11$
8. $B \longleftarrow (B + f(C,D,A) + X_7) \lll 19$ **16.** $B \longleftarrow (B + f(C,D,A) + X_{15}) \lll 19$

Etapa 2:
1. $A \longleftarrow (A + g(B,C,D) + X_0 + P) \lll 3$ **9.** $A \longleftarrow (A + g(B,C,D) + X_2 + P) \lll 3$
2. $D \longleftarrow (D + g(A,B,C) + X_4 + P) \lll 5$ **10.** $D \longleftarrow (D + g(A,B,C) + X_6 + P) \lll 5$
3. $C \longleftarrow (C + g(D,A,B) + X_8 + P) \lll 9$ **11.** $C \longleftarrow (C + g(D,A,B) + X_{10} + P) \lll 9$
4. $B \longleftarrow (B + g(C,D,A) + X_{12} + P) \lll 13$ **12.** $B \longleftarrow (B + g(C,D,A) + X_{14} + P) \lll 13$
5. $A \longleftarrow (A + g(B,C,D) + X_1 + P) \lll 3$ **13.** $A \longleftarrow (A + g(B,C,D) + X_3 + P) \lll 3$
6. $D \longleftarrow (D + g(A,B,C) + X_5 + P) \lll 5$ **14.** $D \longleftarrow (D + g(A,B,C) + X_7 + P) \lll 5$
7. $C \longleftarrow (C + g(D,A,B) + X_9 + P) \lll 9$ **15.** $C \longleftarrow (C + g(D,A,B) + X_{11} + P) \lll 9$
8. $B \longleftarrow (B + g(C,D,A) + X_{13} + P) \lll 13$ **16.** $B \longleftarrow (B + g(C,D,A) + X_{15} + P) \lll 13$
unde s-a folosit constanta $P = 5a827999$.

Etapa 3:
1. $A \longleftarrow (A + s(B,C,D) + X_0 + P) \lll 3$ **9.** $A \longleftarrow (A + s(B,C,D) + X_1 + P) \lll 3$
2. $D \longleftarrow (D + s(A,B,C) + X_8 + P) \lll 9$ **10.** $D \longleftarrow (D + s(A,B,C) + X_9 + P) \lll 9$
3. $C \longleftarrow (C + s(D,A,B) + X_4 + P) \lll 11$ **11.** $C \longleftarrow (C + s(D,A,B) + X_5 + P) \lll 11$
4. $B \longleftarrow (B + s(C,D,A) + X_{12} + P) \lll 15$ **12.** $B \longleftarrow (B + s(C,D,A) + X_{13} + P) \lll 15$
5. $A \longleftarrow (A + s(B,C,D) + X_2 + P) \lll 3$ **13.** $A \longleftarrow (A + s(B,C,D) + X_3 + P) \lll 3$
6. $D \longleftarrow (D + s(A,B,C) + X_{10} + P) \lll 9$ **14.** $D \longleftarrow (D + s(A,B,C) + X_{11} + P) \lll 9$
7. $C \longleftarrow (C + s(D,A,B) + X_6 + P) \lll 11$ **15.** $C \longleftarrow (C + s(D,A,B) + X_7 + P) \lll 11$
8. $B \longleftarrow (B + s(C,D,A) + X_{14} + P) \lll 15$ **16.** $B \longleftarrow (B + s(C,D,A) + X_{15} + P) \lll 15$
unde $P = 6ed9eba1$.

Observaţia 1.2. *Funcţia f mai poate fi definită sub forma*

$$f(X,Y,Z) = \textbf{if } X \textbf{ then } Y \textbf{ else } Z$$

Funcţia $g(X,Y,Z)$ ia (pe biţi) valoarea majoritară dintre valorile (pe biţi) ale lui X, Y, Z.

Exemplul 1.3. *([57]): Câteva amprente obţinute cu ajutorul funcţiei* **MD4***:*

Secvenţă ASCII	*Amprentă (scrisă în hexazecimal)*
""	$31d6cfe0d16ae931b73c59d7e0c089c0$
"a"	$bde52cb31de33e46245e05fbdbd6fb24$
"abc"	$a448017aaf21d8525fc10ae87aa6729d$
"'$abcdefghijklmnopqrstuvwxyz$"	$d79e1c308aa5bbcdeea8ed63df412da9$

1.7.2 Criptanaliza schemei MD4

Etapele funcţiei **MD4** sunt caracterizate de permutări σ_i ale blocului $X = (X_0, \ldots X_{15})$ şi deplasări ciclice spre stânga $\alpha_{i,j}$, $i = 1,2,3$, $j = 1,2,\ldots,16$.
Cele trei permutări – corespunzătoare celor 3 etape – sunt

$$\sigma_1 = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \end{pmatrix}$$
$$\sigma_2 = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 0 & 4 & 8 & 12 & 1 & 5 & 9 & 13 & 2 & 6 & 10 & 14 & 3 & 7 & 11 & 15 \end{pmatrix}$$
$$\sigma_3 = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 0 & 8 & 4 & 12 & 2 & 10 & 6 & 14 & 1 & 9 & 5 & 13 & 3 & 11 & 7 & 15 \end{pmatrix}$$

Deplasările ciclice spre stânga $\alpha_{i,j}$ sunt

$i/j \ (mod\ 4)$	0	1	2	3
1	3	7	11	19
2	3	5	9	13
3	3	9	11	15

De exemplu, $\alpha_{3,6} = 11$.

Funcţiile f şi g au următoarele proprietăţi;

- $f(\mathbf{1}, a, x) = f(\mathbf{0}, x, a) = f(x, a, a) = a, \quad \forall a, x \in Z_2^{32}, \quad \mathbf{0} = 00\ldots0, \ \mathbf{1} = 11\ldots1;$
- $g(a, a, x) = g(a, x, a) = g(x, a, a) = a, \quad \forall a, x \in Z_2^{32};$

În prima etapă σ_1 este permutarea identică, iar X_i sunt utilizate în ordinea iniţială. Deci, dacă $X_0, \ldots, X_{11}$ sunt fixate, se pot alege uşor valori pentru X_{12}, X_{13}, X_{14} astfel ca să se obţină 0 în regiştrii A, C şi D.
Permutarea σ_2 din Etapa 2 asociază pe rând registrului B valorile din $X_{12}, X_{13}, X_{14}, X_{15}$. Deci, dacă celelalte valori sunt $-P$, iar X_{12}, X_{13}, X_{14} sunt alese astfel ca să fie îndeplinită proprietatea anterioară, regiştrii A, C, D rămân cu valoarea $\mathbf{0}$.
Această proprietate are loc pentru orice valoare a lui X_{15}.

Atacul ignoră Etapa 3 şi consideră ieşirea din Etapa 2 ca o funcţie (aleatoare) de variabilă X_{15}, în care trei regiştri sunt permanent $\mathbf{0}$. Deci dimensiunea aleatoare este redusă la 32 biţi; rezultă că un atac bazat pe paradoxul naşterilor poate deduce o coliziune (două blocuri de 32 biţi care dau aceiaşi ieşire după Etapa 2) în aproximativ 2^{16} încercări.

Formal, un atac asupra funcţiei de dispersie **MD4** redusă la primele două etape este:

Intrare: A, B, C, D;

1. **for** $i = 0, 1, 2, 4, 5, 6, 8, 9, 10, 12, 13$ **do** $X_{\sigma_2^{-1}(i)} \longleftarrow -P$
2. $Random(X_{11})$
3. **for** $i = 0$ **to** 11 **do** $(A, D, B, C) \longleftarrow (D, C, B, (A + f(B, C, D) + X_i) \lll \alpha_{1,i})$
4. **for** $i = 12$ **to** 14 **do**
   4.1. $X_i \longleftarrow -(A + f(B, C, D))$
   4.2. $(A, D, C, B) \longleftarrow (D, C, B, \mathbf{0})$
5. $B_0 \longleftarrow A$
6. **if** $u(X_{15}) = u(X_{15}{}')$ **then**
   $X_i{}' = X_i \quad (0 \leq i \leq 14)$
   $Output(X, X'); \quad Exit$

Function $u(X_{15})$:

1. $A \longleftarrow \mathbf{0}, \quad C \longleftarrow \mathbf{0}, \quad D \longleftarrow \mathbf{0}$;
2. $B \longleftarrow ((B_0 + X_{15} + P) \lll \alpha_{1,15})$;
3. **for** $i = 0$ **to** 15 **do**
   $(A, D, C, B) \longleftarrow (D, C, B, (A + g(B, C, D) + X_{\sigma_2(i)} + P) \lll \alpha_{2,i})$
4. $Return(B)$.

După ce se aplică acest atac pentru primele două etape, permutarea σ_3 din Etapa 3 asociază pe X_{15} (singura valoare care se modifică) doar registrului B; deci două blocuri de câte 128 biţi X şi X' (cu $X_i = X_i{}'$ pentru $i = 0, \ldots, 14$, $X_{15} \neq X_{15}{}'$) care duc la o coliziune după primele două etape, au proprietatea că $h(X)$ şi $h(X')$ diferă doar pe al doilea bloc de 32 biţi.
Se pot obţine uşor îmbunătăţiri în care cele două amprente să difere doar într-un singur bit din zona respectivă.

Pe baza acestui atac Hans Dobbertin construieşte un algoritm care generează toate coliziunile din **MD4**. Pentru detalii a se vedea [28] şi [80].

1.7.3 Funcţia de dispersie MD5

Pentru a elimina slăbiciunile sistemului **MD4**, în 1991 Ron Rivest propune o nouă variantă – **MD5**, publicată anul următor sub numele *RFC* 1321 *Internet Standard*.

Schematic, **MD5** se bazează pe o "funcţie de criptare" g propusă de Davies - Meyer: dacă aceasta este

$$g : Z_2^{128} \times Z_2^{512} \longrightarrow Z_2^{128}$$

atunci

$$h(H, X) = H + g(H, X)$$

unde adunarea este realizată modulo 2^{32}.

Criptarea $g(H, X)$ constă din 4 runde şi – în linii mari – poate fi descrisă astfel:

- Mesajul H de 128 biţi este aranjat printr-o permutare (care depinde de rundă) într-o secvenţă de cuvinte A, B, C, D;
- Fiecare din aceste cuvinte trece printr-o transformare secvenţială, bazată pe o schemă de tip Feistel;
- O transformare este definită de un S - box cu intrarea (A, K) (K – cheie de rundă derivată din X) şi ieşirea B, C, D;
- Ieşirea din S box este

$$(A + f_i(B, C, D) + K + k_{i,j} + B) \ll \alpha_{i,j}$$

unde $\alpha_{i,j}$ şi $k_{i,j}$ sunt definite printr-o tabelă, iar f_i este o funcţie booleană de rundă, definită

$$\begin{aligned} f_1(B, C, D) &= (B \wedge C) \vee ((\neg B) \wedge D) \\ f_2(B, C, D) &= (D \wedge B) \vee ((\neg D) \wedge C) \\ f_3(B, C, D) &= B \oplus C \oplus D \\ f_4(B, C, D) &= C \oplus (B \wedge (\neg D)) \end{aligned}$$

Exemplul 1.4. *([57]): Câteva amprente obţinute cu ajutorul funcţiei* **MD5***:*

Secvenţă ASCII	*Amprentă (scrisă în hexazecimal)*
""	$d41d8cd98f00b204e9800998ecf8427e$
"a"	$0cc175b9c0f1b6a831c399e269772661$
"abc"	$900150983cd24fb0d6963f7d28e17f72$
"'$abcdefghijklmnopqrstuvwxyz$"	$c3fcd3d76192e4007dfb496cca67e13b$

Mai multe detalii referitoare la sistemul **MD5** pot fi găsite în [69] sau [80].

Şi această funcţie de dispersie – utilizată mai ales în aplicaţii Internet – este spartă destul de repede. Mai mult, în 2006 V. Klima ([45]) publică un algoritm care calculează – în mai puţin de un minut – coliziuni pentru **MD5**, folosind un PC standard.

1.8 Funcţia de dispersie $SHA1$

SHA1 este o variantă a funcţiei de dispersie SHA – notată ca standard $FIPS\ 180-1$ – care corectează o mică slăbiciune din **SHA**. Construcţia sa este următoarea:

Fie x $(|x| \leq 2^{64}-1)$ şirul binar care trebuie amprentat.
Prima parte a algoritmului este identică cu cea de la **MD4**:

1. $d \longleftarrow (447-|x|)\ (mod\ 512)$
2. $s \longleftarrow$ reprezentarea binară a lui $|x|\ (mod\ 2^{64}), |s| = 64$
3. $M = x\|1\|0^d\|s$

Dacă $|s| < 64$, se adaugă zerouri la stânga până se ajunge la egalitate.
Blocul final M (care intră în algoritmul de dispersie) are o lungime divizibilă cu 512; îl vom scrie ca o concatenare de n blocuri, fiecare de 512 biţi:

$$M = M_1\|M_2\|\ldots\|M_n$$

Fie funcţiile $f_i : Z_2^{32} \times Z_2^{32} \times Z_2^{32} \longrightarrow Z_2^{32} \quad (0 \leq i \leq 79)$, definite astfel:

$$f_t(B,C,D) = \begin{cases} (B \wedge C) \vee ((\neg B) \wedge D) & 0 \leq t \leq 19 \\ B \oplus C \oplus D & 20 \leq t \leq 39 \\ (B \wedge C) \vee (B \wedge D) \vee (C \wedge D) & 40 \leq t \leq 59 \\ B \oplus C \oplus D & 60 \leq t \leq 79 \end{cases}$$

Deci fiecare funcţie f_t ia trei cuvinte la intrare şi scoate un cuvânt la ieşire.

Se mai definesc constantele $K_0, K_1, \ldots, K_{79}$ astfel:

$$K_t = \begin{cases} (5ab27999)_{16} & 0 \leq t \leq 19 \\ (6ed9eba1)_{16} & 20 \leq t \leq 39 \\ (8f1bbcdc)_{16} & 40 \leq t \leq 59 \\ (ca62c1d6)_{16} & 60 \leq t \leq 79 \end{cases}$$

Algoritmul de compresie **SHA1** este:

1. $H_0 \longleftarrow (67452301)_{16}, \quad H_1 \longleftarrow (efcdab89)_{16}, \quad H_2 \longleftarrow (98badcfe)_{16},$
$H_3 \longleftarrow (10325476)_{16}, \quad H_4 \longleftarrow (c3d2e1f0_{16}.$

2. **for** $i \longleftarrow 1$ **to** n **do**

2.1. Fie $M_i = W_0 \| W_1 \| \ldots \| W_{15}, \quad W_i \in Z_2^{32}$.

2.2. **for** $t \longleftarrow 16$ **to** 79 **do**

$$W_t \longleftarrow (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16} \ll 1)$$

2.3. $A \longleftarrow H_0, \quad B \longleftarrow H_1, \quad C \longleftarrow H_2, \quad D \longleftarrow H_3, \quad E \longleftarrow H_4$

2.4. **for** $t \longleftarrow 0$ **to** 79 **do**

2.4.1. $temp \longleftarrow (A \ll 5) + f_t(B, C, D) + E + W_t + K_t$

2.4.2. $E \longleftarrow D,\ D \longleftarrow C,\ C \longleftarrow (B \ll 30),\ B \longleftarrow A$

2.4.3. $A \longleftarrow temp$

2.5 $H_0 \longleftarrow H_0 + A, \quad H_1 \longleftarrow H_1 + B, \quad H_2 \longleftarrow H_2 + C,$
$H_3 \longleftarrow H_3 + D, \quad H_4 \longleftarrow H_4 + E$

3. **Output**: $y = H_0 \| H_1 \| H_2 \| H_3 \| H_4$.

Funcţia de compresie **SHA1** formează pentru fiecare bloc de 512 biţi un bloc de 160 biţi.

Exemplul 1.5. *([57]): Câteva amprente obţinute cu ajutorul funcţiei* **SHA1***:*

Secvenţă ASCII	*Amprentă (scrisă în hexazecimal)*
""	$da39a3ee5e6b4b0d3255bfef95601890afd80809$
"a"	$86f7e437faa5a7fce15d1ddcb9eaeaea377667b8$
"abc"	$a9993e364706816aba3e25717850c26c9cd0d89d$
"'$abcdefghijklmnopqrstuvwxyz$"	$32d10c7b8cf96570ca04ce37f2a19d84240d3a89$

Algoritmul original **SHA** avea o slăbiciune[4] care permitea aflarea coliziunilor în aproximativ 2^{61} paşi. Versiunea **SHA1** este mai eficientă, atacul naşterilor conducând la aflarea coliziunilor abia după 2^{80} paşi. Dar şi pentru această funcţie de dispersie au început să fie publicate coliziuni începând cu 2005, când o echipă de la Universitatea Shandong, China afirmă că a găsit coliziuni pentru **SHA1** în 2^{69} operaţii (iar pentru o variantă **SHA1** de 58 runde, în numai 2^{33} operaţii) .

[4]Diferenţa dintre **SHA** şi **SHA1** este minoră: în **SHA** transformarea de la pasul 2.2 se efectuează fără nici o rotaţie !

La 30 mai 2001 $NIST$ propune o nouă versiune **SHA2**; aceasta include **SHA1** precum şi alte trei funcţii de dispersie $\mathbf{SHA_{256}}$, $\mathbf{SHA_{384}}, \mathbf{SHA_{512}}$ (cifrele se referă la mărimea amprentei).

În momentul scrierii acestei cărţi este lansat un concurs pentru un nou standard de dispersie care va fi numit **SHA3**. Detalii asupra ultimelor rezultate se pot găsi la

$$http://ehash.iaik.tugraz.at/wiki/The\_SHA-3\_Zoo$$

1.9 Funcţii de dispersie aleatoare

Deşi satisface **CP**, o funcţie de dispersie criptografică h poate dezvălui unele informaţii despre mesajele amprentatate.
De exemplu, într-un sistem multi-user, parolele utilizatorilor sunt amprentate cu o anumită funcţie de dispersie, iar aceste amprente sunt stocate într-un fişier.
Cât de sigur este acesta ?

Oscar poate accesa – într-o modalitate oarecare – fişierul cu parole şi găseşte amprentele tuturor parolelor.

- Multe parole sunt cuvinte sau propoziţii din limbajul natural. *Oscar* poate crea (anterior atacului) o tabelă cu amprente ale tuturor cuvintelor din dicţionar, după care compară amprentele din fişier cu această tabelă. Este aşa numitul *dictionary attack*.

- Dacă două persoane au aceeaşi parolă (lucru posibil frecvent), *Oscar* va dispune de această informaţie (deoarece h este o funcţie deterministă).

O soluţie la această problemă constă în utilizarea de factori aleatori în construirea funcţiilor de dispersie. Anume, odată cu crearea unei parole este generat şi un număr aleator r; acesta este amprentat odată cu parola. Numărul r este păstrat petru verificare.
De exemplu

$$h(x) = (r, MD5(r, x)) \quad \text{sau} \quad h(x) = (r, r^{MD5(x)})$$

În acest fel se elimină multe din problemele generate de utilizarea frecventă a unei singure funcţii de dispersie.

1.10 Clase de dispersie tari universale

Aceste clase de funcţii de dispersie sunt utilizate pe scară largă în diverse arii criptografice. Să începem cu o definiţie:

Definiţia 1.4. *Fie $(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$ o (N, M) - clasă de dispersie. Ea este "tare universală" dacă*

$$(\forall x, x' \in \mathcal{X})(\forall y, y' \in \mathcal{Y})(x \neq x') \Longrightarrow card\left(\{K \in \mathcal{K} \mid h_K(x) = y,\ h_K(x') = y'\}\right) = \frac{card(\mathcal{K})}{M^2}$$

Exemplul 1.6. *Fie*

$$\mathcal{X} = \mathcal{Y} = Z_3, \quad \mathcal{K} = Z_3 \times Z_3$$

Pentru fiecare $K = (a, b) \in \mathcal{K}$ *şi* $x \in \mathcal{X}$ *definim*

$$h_{a,b}(x) = ax + b \pmod 3$$

Fie

$$\mathcal{H} = \{h_{a,b} \mid (a, b) \in \mathcal{K}\}.$$

Se verifică uşor că $(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$ *formează o clasă de dispersie tare universală.*

Lema 1.2. *Fie* $(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$ *o* (N, M) *- clasă de dispersie tare universală. Atunci*

$$(\forall x \in \mathcal{X})(\forall y \in \mathcal{Y}) \left(card\left(\{K \in \mathcal{K} \mid h_K(x) = y\}\right) = \frac{card(\mathcal{K})}{M} \right).$$

Demonstraţie. Fie $x, x' \in \mathcal{X},\ x \neq x'$ şi $y \in \mathcal{Y}$. Atunci
$card\left(\{K \in \mathcal{K} \mid h_K(x) = y\}\right) = \sum_{y' \in \mathcal{Y}} card(\{K \in \mathcal{K} \mid h_K(x) = y,\ h_K(x') = y'\}) =$
$= \sum_{y' \in \mathcal{Y}} \frac{card(\mathcal{K})}{M^2} = \frac{card(\mathcal{K})}{M}.$ □

Pe baza acestei leme se pot defini diverse construcţii de clase de dispersie tari universale. Astfel:

Teorema 1.6. *Fie* q *un număr prim. Pentru* $a, b \in Z_q$ *definim funcţia* $f_{a,b} : Z_q \longrightarrow Z_q$ *prin*

$$f_{a,b}(x) = ax + b \pmod q$$

Atunci $(Z_q, Z_q, Z_q \times Z_q, \{f_{a,b} \mid a, b \in Z_q\})$ *este o clasă de dispersie tare universală.*

Demonstraţie. De remarcat că această teoremă generalizează Exemplul 1.6.
Fie $x, x', y, y' \in Z_q$ cu $x \neq x'$. Trebuie arătat că există o cheie unică $(a, b) \in Z_q \times Z_q$ cu $ax + b = y,\ ax' + b = y'$ (calculele sunt efectuate modulo q). Aceasta este soluţia (unică) a sistemului de două ecuaţii liniare, având ca necunoscute $a, b \in Z_q$. Mai exact

$$a = \frac{y' - y}{x' - x} \pmod q, \qquad b = y - x \cdot \frac{y' - y}{x' - x} \pmod q$$

□

Teorema 1.7. *Fie* p *un număr întreg pozitiv şi* q *un număr prim.*
Definim $\mathcal{X} = Z_2^p \setminus \{0, 0, \ldots, 0)\}$.

Pentru fiecare $\mathbf{r} \in Z_q^p$ *definim*

$$f_{\mathbf{r}}(\mathbf{x}) = \mathbf{r} \cdot \mathbf{x} \pmod q$$

unde $\mathbf{r} \cdot \mathbf{x}$ *este produsul scalar al vectorilor* $\mathbf{r}$ *şi* $\mathbf{x}$*, calculat modulo* q.

Atunci $(\mathcal{X}, Z_q, Z_q^p, \{f_{\mathcal{R}} \mid \mathcal{R} \in Z_q^p\})$ *este o clasă de de dispersie tare universală.*

Demonstraţie. Fie $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$, $\mathbf{x} \neq \mathbf{x}'$ şi $y, y' \in Z_q$. Trebuie arătat că numărul vectorilor $\mathbf{r} \in Z_q^p$ astfel ca $\mathbf{r} \cdot \mathbf{x} = y \ (mod\ q)$, $\mathbf{r} \cdot \mathbf{x}' = y' \ (mod\ q)$ este constant. Vectorii $\mathbf{r}$ căutaţi sunt soluţia sistemului de două ecuaţii liniare cu p necunoscute peste Z_q. Cele două ecuaţii sunt liniar independente; deci numărul de soluţii ale sistemului este constant: q^{p-2}. □

1.11 Exerciţii

1.1. *Fie $h : \mathcal{X} \longrightarrow \mathcal{Y}$ o (N, M) - funcţie de dispersie. Pentru orice $y \in Y$ notăm $h^{-1}(y) = \{x \mid h(x) = y\}$ şi $s_y = |h^{-1}(y)|$. Fie*

$$S = card(\{\{x_1, x_2\} \mid x_1 \neq x_2,\ h(x_1) = h(x_2)\}).$$

S este numărul perechilor din $\mathcal{X}$ care formează o coliziune pentru h.

- *Arătaţi relaţia $\sum_{y \in \mathcal{Y}} s_y = N$.*
  Astfel, se poate defini media elementelor s_y prin $\mathbf{s} = \frac{N}{M}$.
- *Arătaţi relaţia $S = \sum_{y \in \mathcal{Y}} C_{s_y}^2 = \frac{1}{2} \sum_{y \in \mathcal{Y}} s_y^2 - \frac{N}{2}$.*
- *Arătaţi relaţia $\sum_{y \in \mathcal{Y}} (s_y - \mathbf{s})^2 = 2S + N - \frac{N^2}{M}$.*
- *Deduceţi inegalitatea $S \leq \frac{1}{2}\left(\frac{N^2}{M} - N\right)$.*

  Arătaţi ca avem egalitate dacă şi numai dacă $s_y = \frac{N}{M}$ pentru orice $y \in \mathcal{Y}$.

1.2. *Fie $p = 15083, \alpha = 154, \beta = 2307$ parametrii pentru funcţia de dispersie Chaum - van Heijst - Pfitzmann. Fiind dată coliziunea $\alpha^{7431}\beta^{5564} \equiv \alpha^{1459}\beta^{954} \ (mod\ p)$, calculaţi $log_\alpha \beta$.*

1.3. *([37]) Fie $n = pq$ unde p, q sunt numere prime distincte (secrete) astfel încât $p = 2p_1 + 1,\ q = 2q_1 + 1,\ p, q$ numere prime. Fie $\alpha \in Z_n^*$ un element de ordin $2p_1q_1$ (acesta este ordinul maximal în Z_n). Se defineşte funcţia de dispersie $h : \{1, \ldots, n^2\} \longrightarrow Z_n^*$ prin*

$$h(x) = \alpha^x \ (mod\ n).$$

Să presupunem $n = 603241,\ \alpha = 11$ şi că s-a găsit tripla coliziune

$$h(1294755) = h(80115359) = h(52738737).$$

Utilizaţi această informaţie pentru a factoriza n.

1.4. *Fie $h_1 : Z_2^{2m} \longrightarrow Z_2^m$ o funcţie de dispersie cu coliziuni tari.*

- *Fie* $h_2 : Z_2^{4m} \longrightarrow Z_2^m$ *definită prin regulile:*
  - *scrie* $x \in Z_2^{4m}$ *sub forma* $x = x_1 \| x_2,\ x_1, x_2 \in Z_2^{2m}$*;*
  - *defineşte* $h_2(x) = h_1(h_1(x_1) \| h_1(x_2))$.

  Arătaţi că h_2 *este cu coliziuni tari.*

- *Pentru orice număr întreg* $i \geq 2$ *se defineşte funcţia de dispersie* $h_i : Z_2^{2^i m} \longrightarrow Z_2^m$ *definită recursiv prin regulile:*
  - *scrie* $x \in Z_2^{2^i m}$ *sub forma* $x = x_1 \| x_2,\ x_1, x_2 \in Z_2^{2^{i-1} m}$*;*
  - *defineşte* $h_i(x) = h_1(h_{i-1}(x_1) \| h_{i-1}(x_2))$.

  Arătaţi că h_i *este cu coliziuni tari.*

1.5. *Fie* $f : Z_2^m \longrightarrow Z_2^m$ *o funcţie pentru care problema* **CSP** *este satisfăcută. Definim funcţia de dispersie* $h : Z_2^{2m} \longrightarrow Z_2^m$ *astfel: dacă* $x_1, x_2 \in Z_2^m$ *fie* $x = x_1 \| x_2$ *şi*

$$h(x) = f(x_1 \oplus x_2)$$

Arătaţi că h *nu satisface problema* **CSP**.

1.6. *Fie* q *un număr prim impar. Pentru* $a, b \in Z_q$ *definim* $f_{a,b} : Z_q \longrightarrow Z_q$ *prin*

$$f_{a,b}(x) = (x + a)^2 + b \pmod q$$

Demonstraţi că $(Z_q, Z_q, Z_q \times Z_q, \{f_{a,b} \mid a, b \in Z_q\})$ *este o* (q, q) *- clasă de dispersie tare universală.*

Capitolul 2

Protocoale de autentificare

Procesul de autentificare este un mecanism frecvent utilizat în toate mediile. Orice comunicare începe prin stabilirea identificării partenerilor: fiecare trebuie să prezinte dovezi care să îl autentifice în faţa celuilalt (sau în faţa unui sistem care îi acordă acces la anumite resurse).

Exemplul 2.1. *Istoria cunoaşte numeroase exemple de mecanisme de autentificare. Astfel, unele documente arată că în China de acum 2000 ani, amprentele formau un mijloc de identificare.*
Mesagerii regali din Evul Mediu foloseau ca mjloc de identificare peceţile inelelor.
Sistemul Bertillon (datând din 1870) a fost utilizat în SUA pentru identificarea criminalilor. Acest sistem identifica o persoană printr-un ansamblu de măsurători cum ar fi înălţimea, lungimea braţelor, grosimea toracelui, lungimea degetului inelar, mărimea piciorului, circumferinţa capului. După ce în 1903 – într-un caz celebru – sunt identificaţi doi criminali cu aceleaşi măsuri Bertilion, sistemul a fost înlocuit cu identificarea prin amprente.

Mecanismele de autentificare se referă atât la partenerii de dialog cât şi la mesajele transmise de aceştia.

Principala deosebire între cele două tipuri constă în faptul că autentificarea unui mesaj nu dă nici o garanţie asupra momentului creerii mesajului, pe când autentificarea unui utilizator este implicit legată de momentul solicitării acestei autentificări. Analog, autentificarea unui utilizator nu oferă nici o informaţie despre conţinutul mesajelor pe care le va gestiona acesta în calitate de entitate autorizată, pe când autentificarea unui mesaj oferă anumite date despre conţinutul mesajului.

Aproape toate protocoalele prezentate în capitolele următoare se vor referi – implicit sau explicit – la autentificări de mesaje sau de utilizatori care solicită acces la anumite drepturi.

2.1 Autentificarea utilizatorului

Sunt mai multe motive pentru care este necesară o autentificare a utilizatorului:

Controlul accesului. Tendinţa actuală este de a întări complexitatea drepturilor de acces la resursele accesibile prin reţele de calculatoare (inclusiv Internet). Controlul accesului acordă sau restricţionează dreptul unui utilizator de a accesa anumite resurse; de asemenea, el protejează resursele, limitând accesul doar pentru utilizatorii autorizaţi. Cu ajutorul calculatoarelor se poate asigura un control al accesului pentru diverse tipuri de conexiuni, pentru unele baze de date sau chiar pentru intrarea în anumite clădiri.

Autorizarea. Autorizarea este procesul prin care unui utilizator i se alocă diverse drepturi de acces. Aceste drepturi includ anumite specificaţii, cum ar fi dreptul de a citi, de a scrie sau de a actualiza un anumit fişier. De exemplu, protocoalele de plăţi electronice solicită autorizarea celor care le utilizează, pentru a preveni fraudarea tranzacţiilor. O solicitare similară apare şi în cazul protocoalelor de vot electronic.

Auditarea. Chiar dacă nu sunt prevăzute protocoale de control, este adesea recomandabil să se reţină acţiunile tuturor utilizatorilor. Astfel de arhive sunt utile în cazul unor posibile apariţii de activităţi maliţioase. Experienţa arată multe atacuri asupra sistemelor de calcul provin de la persoane autorizate chiar de sistemele respective.

Este interesantă şi situaţia reciprocă: componentele electronice trebuie de asemenea autentificate. Un exemplu simplu: în unele locaţii există ATM-uri (cititoare de carduri) false; utilizatorii merg la ele, dau cardul şi PIN-ul, iar maşina le spune că momentan nu îi poate servi. La sfârşitul zilei, baza de date construită cu informaţiile de pe carduri şi PIN-urile respective este extrasă din maşină şi utilizată pentru furtul banilor din conturi.

Observaţia 2.1. *În general există o distincţie netă între noţiunile de "autentificare" a utilizatorului şi "identificarea" sa. Protocoalele de identificare se referă la situaţia următoare: utilizatorul respectiv a fost autentificat dar el trebuie să demonstreze că are dreptul să solicite accesul la anumite resurse. Vom întâlni astfel de instrumente de identificare în cadrul capitolelor legate de protocoale electronice de plată sau de vot electronic.*
De asemenea, noţiunile numite generic "zero - knowledge" se referă la protocoale teoretice de identificare.

În general, sistemele de autentificare a utilizatorilor se construiesc cu scopul de a determina

1. Ceva ce utilizatorul **ştie**. De exemplu, o *parolă*.

   Multe sisteme solicită CNP-ul sau numele de fată al mamei pentru a autentifica un utilizator.

2. Ceva ce utilizatorul **posedă**. De obicei este vorba de un lucru – fizic sau electronic – care aparţine utilizatorului. Cheile obişnuite de la uşă sunt exemple tipice. Dar aici poate fi inclus şi un anumit soft de firmă.

3. Ceva ce utilizatorul **este** (sau îl caracterizează). Acesta este principiul *biometric* de măsurare a anumitor proprietăţi biologice ale utilizatorului. Aceste proprietăţi biometrice pot fi statice (măsurători de amprente, retină etc) sau dinamice (analiza vocii, recunoaşterea scrisului de mână etc)

2.1.1 Parole

Cele mai răspândite mijloace de autentificare sunt *parolele (passwords)*. O parolă este o secvenţă alfanumerică cunoscută doar de utilizator (care se autentifică) şi de sistemul care asigură autentificarea. Uneori este posibil ca sistemul să nu cunoască parola, dar să o poată deduce din anumite informaţii pe care le deţine despre utilizator.

În mod uzual, scenariul este următorul: *Alice* are nevoie să acceseze resursele unui sistem (de exemplu un cont, o imprimantă, o aplicaţie soft). Pentru aceasta, ea trimite sistemului o pereche *(Alice, parolă)* şi – explicit sau implicit – specifică o resursă. Parola este folosită pentru a întări identitatea lui *Alice*; similar cu cererea ca la intrarea într-o instituţie, să prezinţi un act de identitate care să ateste cine declari că eşti !). La recepţionarea perechii, sistemul verifică parola şi – dacă ea corespunde celei asociate lui *Alice* – autorizează accesul la resursă[1].

Evident, din motive de securitate, parolele trebuie să fie secvenţe pe care *Alice* le poate memora sau – eventual – deduce uşor. În acelaşi timp, ele trebuie să fie greu de "ghicit" de către orice altă entitate exterioară. Aceste proprietăţi sunt oarecum contradictorii: ceva ce este uşor de memorat are entropie mică, deci este şi uşor de ghicit.

De asemenea, sistemul păstrează parolele sub o formă criptată sau ca amprente (folosind un standard de dispersie), pentru a evita dezvăluirea lor în cazul unui atac reuşit asupra sistemului. Un exemplu sugestiv de gestiune a parolelor a fost prezentat în [2], pag. 71-72 referitor la parolele UNIX.

Vulnerabilităţi ale parolelor

Într-un survey publicat în 1979 în Communication of the ACM (pag 594-597), Morris şi Thompson au anallizat 3289 parole colectate aleator de la angajaţii care utilizau tehnică de calcul (în special IBM) şi au găsit că 2831 (86%) din acesta erau vulnerabile deoarece:

[1]Pentru unele resurse este asignat şi un "tichet", care acordă un acces limitat în timp.

15	erau formate dintr-un singur caracter ASCII
72	erau formate din 2 caractere ASCII
464	erau formate din 3 caractere ASCII
477	erau formate din 4 caractere alfanumerice
706	erau formate din 5 litere, de acelaşi tip (mari sau mici)
605	erau formate din 6 litere, toate mici.
	(de remarcat că $26^6 = 309M$, o dimensiune relativ mică).

Alte vulnerabilităţi ale parolelor includ în general:
- *Cuvinte uzuale.* Engleză, română sau alte limbi.
- *Nume cunoscute.* Nume de vedete, animale, prieteni, membri ai familiei, porecle.
- *Informaţii uşor de obţinut.* Date de naştere, numere de telefon, numărul maşinii etc.
- *Secvenţe de tastatură.* Ceva de genul "qwerty".
- *Permutări ale celor de sus.* De obicei scrieri inverse (în oglindă).

Sunt diverse recomandări privind construcţia şi păstrarea parolelor. De asemenea, securitatea unui sistem este mai bună dacă parolele se schimbă frecvent (cel puţin odată pe an).

Parole one-time

Parolele *one - time* sunt parole care se folosesc o singură dată. Acest procedeu asigură o mai bună securitate în faţa atacurilor prin forţă brută. Există diverse strategii de a asigura astfel de parole. Astfel:

- Unele sisteme oferă utilizatorului o listă de parole. Atunci când utilizatorul foloseşte o parolă, sistemul verifică dacă este în această listă. În caz afirmativ, autentifică intrarea şi – în paralel – elimină parola din listă.
  O variantă foloseşte o tabelă în care elementele sunt perechi *întrebare/răspuns*. Aici utilizatorul solicită autentificarea, sistemul alege aleator o întrebare şi aşteaptă răspunsul. Dacă acest răspuns corespunde întrebării respective, utilizatorul este autentificat, iar sistemul elimină perechea respectivă din tabelă (deci ulterior acea întrebare nu mai este folosită).

- Parole actualizate secvenţial: iniţial există o singură parolă (secretă). În timpul autentificării folosind parola α, utilizatorul generează şi transmite sistemului o nouă parolă $\alpha' = e_K(\alpha)$, unde K este o cheie derivată din α. Pentru următoarea comunicare, α este înlocuită cu α'. Metoda – deşi promiţătoare – devine dificilă atunci când apar întreruperi de comunicaţie.

- Parole bazate pe funcţii neinversabile. Par cele mai eficiente tipuri de parole *one - time*. Cea mai cunoscută strategie de construcţie de astfel de parole este schema Lamport.

În acest protocol este folosită o funcţie neinversabilă ϕ. În faza de iniţializare a schemei, *Alice* efectuează următoarele operaţii:

1. Alege un mesaj secret w şi un număr n de autentificări bazate pe w (uzual $n = 100$ sau $n = 1000$).
2. Transferă lui *Bob* printr-un canal sigur valoarea $w_0 = \phi^n(w)$.
3. *Bob* iniţializează un counter $i_{Alice} := 1$.

În timpul autentificării pentru deschiderea sesiunii cu numărul i, se procedează astfel:

1. *Alice* calculează $w_i = \phi^{n-i}(w)$ şi trimite lui *Bob* tripletul
$$(Alice, i, w_i)$$
(calculul lui w_i se poate face la fiecare nouă sesiune, sau se poate deduce dintr-o tabelă construită iniţial, odată cu operaţiile efectuate la determinarea lui w_0).
2. *Bob* verifică dacă $i = i_{Alice}$ şi dacă $\phi(w_i) = w_{i-1}$. Dacă ambele relaţii sunt verificate, *Bob* acceptă autentificarea, salvează w_i pentru verificarea următoare şi incrementează contorul: $i_{Alice} := i_{Alice} + 1$.

Ca o variantă a protocolului Lamport, *Alice* poate deţine o parolă α dată de *Bob*. Atunci când doreşte autentificarea, *Alice* trimite o pereche $(r, h(r\|\alpha))$, unde r este o secvenţă binară, iar h este o funcţie de dispersie. *Bob* face verificarea calculând $h(r\|\alpha)$ şi comparând rezultatul cu al doilea element al perechii primite.
Pentru a elimina atacurile unui adversar activ, r trebuie să fie un element de tip *nonce*[2].

2.1.2 Măsurători biometrice

Măsurătorile biometrice folosesc pentru autentificare caracteristicile fizice ale unei persoane. În general ele nu sunt folosite drept parole, deoarece – odată compromise, nu pot fi înlocuite. Cele mai utilizate măsurători biometrice utilizate pentru autentificarea unei persoane sunt: Recunoaşterea vocii, Dinamica semnăturii, Amprente, Geometria mâinii, Scanarea retinei, Scanarea irisului, Modelul facial sau alte caracteristici specifice.

La construirea unui sistem de autentificare bazat pe măsurători biometrice trebuie ţinut cont de timpul necesar efectuării acestor măsurători, de preţul aparaturii, de gradul de acceptare al utilizatorului de a se expune măsurătorilor biometrice, de ratele de eroare privind falsa - acceptare (a unui alt utilizator decât cel legal) şi falsa - respingere (rejectarea utilizatorului legal); de obicei, aceste două rate se consideră egale, deşi în realitate ele depind în primul rând de gradul de performanţă al aparaturii folosite.

[2]Număr generat aleator şi folosit o singură dată.

Un studiu realizat în 2000 de Sandia National Labs (SUA) compară aceste modalităţi de autentificare a utilizatorului. Pe scurt, ele pot fi sumarizate în tabelele următoare:

Tehnica	**Rata de eroare**
Voce (analiză Alpha)	3%
Voce (analiză ECCO)	2%
Semnătură	2%
Scanarea retinei	0.4%
Geometria mâinii	0.1%
Amprente	9% falsa - respingere, 0% falsa - acceptare

Caracteristici	**Cea mai bună**	**Cea mai slabă**
Acceptarea utilizatorului	Mâna	Voce
Falsa - respingere	Mâna	Amprente
Falsa - acceptare	Mâna, retina, amprente	Voce
Mulţime detalii	Mâna, retina, amprente	Voce, semnătura
Dificultatea imitării	Retina	Voce, semnătura
Cost	Voce	Retina

Datorită ratei mari de eroare, măsurătorile biometrice nu constituie un avantaj faţă de parole. De obicei aceste două tipuri de autentificare se folosesc combinat (deci sistemele verifică atât ceea ce utilizatorul "*este*" cât şi ceea ce utilizatorul "*ştie*").

În secţiunile următoare ale acestui capitol ne vom ocupa de protocoale de autentificare a mesajelor. Vom face acest lucru separat de autentificarea utilizatorului care trimite mesajul (autentificarea ambelor entităţi se realizează folosind semnătura electronică, modalitate care va fi prezentată în capitolul următor).

De asemenea, autentificarea unui mesaj nu înseamnă totdeauna şi integritatea lui (deşi în majoritatea protocoalelor, acest lucru se subînţelege).

Sunt domenii de securitate a informaţiei – de care ne vom ocupa în alte volume – dedicate special problemelor de autentificare şi integritate a mesajelor. Amintim în acest sens Watermarking şi Fingerprint. Acum, în această etapă, vom discuta doar autentificarea prin Coduri de Autentificare a Mesajelor (MAC) şi prin canale subliminale.

2.2 MAC

Un cod de autentificare a mesajului (*MAC – Message Authentication Code*) este o combinaţie între o funcţie de compresie şi o cheie. El este folosit pentru a autentifica mesajul, asigurând în acelaşi timp şi certificarea integrităţii sale.
Codurile de autentificare a mesajelor sunt utilizate pe scară largă în protocoale legate de comunicarea pe Internet, cum sunt IPSec sau SSL/TLS.

Atunci când *Alice* trimite lui *Bob* un mesaj α (criptat sau nu), ea va construi un cod $MAC(\alpha)$ folosind o cheie şi funcţie de compresie cunoscute de ambii parteneri, după care va trimite perechea

$$(\alpha, MAC(\alpha))$$

La recepţia unei perechi (α, x), *Bob* calculează $MAC(\alpha)$ şi vede dacă acesta coincide cu x. În caz afirmativ, el va considera mesajul α ca fiind autentic.

Definiţia 2.1. *O pereche* (α, x) *cu* $x = MAC(\alpha)$ *se numeşte "pereche validă".*

De remarcat că atât *Alice* cât şi *Bob* pot calcula un MAC valid pentru un mesaj α.

2.2.1 $HMAC$

Dacă drept funcţii de compresie sunt folosite funcţii de dispersie, codurile de autentificare a mesajelor se numesc coduri $HMAC$ (Hash MAC).

Exemplul 2.2. *Unul din primele HMAC-uri a fost propus de IBM pentru mesajele trimise pe Internet. El este construit în felul următor:*
Fie K *cheia secretă folosită; ea este partajată în două subchei* $K = (k_1, k_2)$.
Pentru fiecare bloc de text clar α*, codul de autentificare va fi*

$$MAC(\alpha) = MD5(k_1 \| MD5(k_2 \| \alpha))$$

unde $MD5$*este funcţia de dispersie prezentată în Capitolul* 1.

Codul $HMAC$ prezentat în continuare a fost construit de M. Bellare, R. Canetti şi H. Krawczyk, fiind cunoscut drept standardul RFC 2104 ([47]). El este de fapt o extensie a codului din Exemplul 2.2

Fie h o funcţie de dispersie criptografică care procesează mesaje de n octeţi şi produce rezumate de p octeţi (dacă h este $SHA1$ atunci $n = 64$ şi $p = 20$).

Se mai defineşte un parametru t $(4 < t < p)$ care reprezintă numărul de octeţi din $HMAC$.
Dacă x este blocul de intrare şi K este cheia folosită, calculul lui $HMAC(x)$ este realizat de algoritmul următor:

1. Dacă $|K| > 8n$, se înlocuieşte K cu $h(K)$.
2. Se completează K la dreapta cu biţi $'0'$ până ajunge la n octeţi.
3. Se calculează
$$A = h(K \oplus opad \| h(K \oplus ipad \| x))$$
unde $ipad, opad \in \{0,1\}^{8n}$, $ipad = (36 \ldots 36)_{16}$, $opad = (5C \ldots 5C)_{16}$.
4. $HMAC_K(x)$ este format din primii t octeţi din A.

Pentru a analiza securitatea acestui cod de autentificare, vom defini noţiunea de *MAC sigur* (pentru mesaje de o anumită lungime).

Este evident că obiectivul unui atac (produs de *Oscar*) este de a produce o pereche (α, x) validă pentru o cheie K (necunoscută dar fixată).
Printr-un atac cu text clar ales, *Oscar* va solicita MAC-uri pentru mesajele $\alpha_1, \alpha_2, \ldots, \alpha_n$ alese de el. Teoretic, putem considera un oracol ("black box") care răspunde la cererile lui *Oscar*, dând MAC-urile respective.

Deci, acesta va dispune de o listă de perechi valide

$$(\alpha_1, x_1), (\alpha_2, x_2), \ldots (\alpha_n, x_n)$$

generate cu o cheie necunoscută K.

Ulterior, când *Oscar* va emite o pereche (α, x), se presupune că $x \notin \{x_1, x_2, \ldots, x_n\}$. Dacă perechea emisă este validă, atunci spunem că este un *fals*.

Un MAC pentru care probabilitatea de a obţine un fals este neglijabilă se numeşte *MAC sigur*.

Pe baza acestor noţiuni, securitatea codului HMAC construit anterior este demonstrată cu ajutorul rezultatului următor:

Teorema 2.1. *Fie $h : \{0,1\}^* \longrightarrow \{0,1\}^p$ o funcţie de dispersie criptografică. Fiind date două chei $K_1, K_2 \in \{0,1\}^p$, considerăm algoritmul MAC definit prin*

$$MAC_{K_1,K_2}(x) = h(K_2 \| h(K_1 \| x))$$

Dacă aplicaţia MAC_{K_2} definită $MAC_{K_2}(x) = h(K_2 \| x)$ este un algoritm MAC sigur pentru mesaje x, $|x| = p$, atunci MAC_{K_1,K_2} este un algoritm MAC sigur pentru mesaje de lungime arbitrară.

Demonstraţie. ([80]). Să presupunem că avem un oracol $\mathcal{O}$ care dă valoarea $MAC_{K_2}(x)$ pentru orice x cu $|x| = p$, şi un adversar *Oscar* pentru MAC_{K_1,K_2}; vom construi un adversar pentru MAC_{K_2} în felul următor:

1. Generăm aleator K_1 şi simulăm un oracol pentru *Oscar*.

2. De câte ori *Oscar* trimite spre oracol un α_i, calculăm $h(K_1 \| \alpha_i)$ şi-l trimitem spre $\mathcal{O}$; acesta dă un răspuns x_i, pe care îl returnăm lui *Oscar* ca răspuns la cererea sa. Aceste procedeu se repetă pentru $i = 1, 2, \ldots, n$ (n o valoare arbitrară).

3. Când *Oscar* termină anunţând că a găsit o pereche falsă (α, x), calculăm $h(K_1 \| \alpha)$ şi-l trimitem spre $\mathcal{O}$. Dacă acesta răspunde cu una din valorile $x_1, \ldots, x_n$ înseamnă că am găsit o coliziune pentru funcţia de dispersie h – lucru imposibil, deoarece aceasta este o funcţie criptografică (deci cu coliziuni tari). Rezultă că $h(K_1 \| \alpha)$ este diferit de toate valorile $h(K_1 \| \alpha_i)$ calculate anterior. Notând $\beta = h(K_1 \| \alpha)$, tragem concluzia că am găsit o pereche falsă (β, x) cu $|\beta| = p$; lucru imposibil deoarece MAC_{K_2} este – prin ipoteză – un MAC sigur pentru mesaje de lungime p.

□

2.2.2 $CBC - MAC$

Cea mai simplă modalitate de a construi un cod de autentificare al unui mesaj α este de a folosi un sistem de criptare simetric implementat în modul CBC şi de a utiliza ultimul bloc obţinut prin criptarea lui α drept $MAC(\alpha)$.
Metoda este numită $CBC - MAC$ şi a fost prezentată în [2], pag. 71 (folosind sistemul DES). Să o reamintim:

1. Textul clar x se împarte în blocuri de lungime fixată: $x = \alpha_1\alpha_2 \dots \alpha_n$.
2. Se construieşte secvenţa $\beta_1, \beta_2, \dots, \beta_n$ după formula
$$\beta_i = e_K(\beta_{i-1} \oplus \alpha_i) \qquad (i \geq 1)$$
unde $\beta_0 = 00 \dots 0$, iar e_K este funcţia de criptare cu cheia K.
3. $MAC(x) = \beta_n$.

Metoda este foarte rapidă şi uşor de realizat. Ea prezintă însă unele slăbiciuni.
Astfel, să presupunem că ştim trei perechi valide (x_1, c_1), (x_2, c_2), (x_3, c_3). Mai mult, x_1 şi x_3 sunt mesaje de aceeaşi lungime, iar x_1 este un prefix propriu al lui x_2; deci putem scrie $x_2 = x_1\|\alpha\|x_2'$, unde α este un bloc.

Blocul criptat obţinut – în calculul lui c_2 – după criptarea lui α este $e_K(\alpha \oplus c_1)$.

Să notăm $\alpha' = \alpha \oplus c_1 \oplus c_3$ şi $x_4 = x_3\|\alpha'\|x_2'$.

În calculul MAC-ului c_4 (pentru x_4), după criptarea lui α' se obţine
$$e_K(c_3 \oplus \alpha') = e_K(\alpha \oplus c_1),$$
după care criptările pentru c_2 şi c_4 merg similar (urmând după blocul α'). Deci, în final $c_4 = MAC(x_4) = c_2$ şi se poate construi o pereche validă (x_4, c_2).

O variantă este codul de autentificare $EMAC$ (*Encrypted MAC*), obţinut prin criptarea cu altă cheie K' a ultimului bloc criptat β_n.
În acest caz, securitatea este egală cu rezistenţa lui CBC la un atac bazat pe paradoxul naşterilor.
Într-adevăr, să presupunem că *Oscar* obţine o coliziune: două perechi valide (x_1, c), (x_2, c) cu acelaşi MAC. El ia un mesaj arbitrar x_3' şi solicită un cod de autentificare pentru mesajul $x_3 = x_1\|x_3'$. Să presupunem că primeşte $MAC(x_3) = c'$. Atunci $(x_2\|x_3', c')$ este o pereche validă.

În [80] se arată că probabilitatea de a obţine o coliziune din $t \cdot \sqrt{N}$ perechi valide (unde N este numărul de MAC-uri posibile) este $1 - e^{-t^2/2}$.

Deci – pentru a rezista la un astfel de atac – modul CBC de implementare utilizat pentru construirea unui $EMAC$ trebuie să fie similar unei funcţii de dispersie criptografică.

O modalitate simplă de a realiza acest lucru constă în eliminarea unor biţi din MAC; de exemplu trunchierea rezultatului la prima jumătate; acesta este de fapt standardul ISO/IEC 9797 (stabilit în 1989) de obţinere a codului de autentificare bazat pe algoritmi de criptare simetrici.

OMAC

OMAC (*One-key CBC MAC*) este un standard care operează cu mesaje a căror lungime nu este obligatoriu un multiplu al lungimii blocului de criptare. El lucrează cu un sistem simetric $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$, având lungimea blocurilor criptate egală cu p, o cheie $K \in \mathcal{K}$, o familie $\mathcal{A}$ de constante şi două constante $C_1, C_2 \in \mathcal{A}$. Se mai defineşte o valoare t $(t < p)$ ca fiind lungimea codului de autentificare.
Fie familia de funcţii

$$\mathcal{H} = \{H_L \mid L \in \mathcal{C}, \;\; H_L : \mathcal{A} \longrightarrow \mathcal{C}\}$$

Să considerăm textul clar $x = \alpha_1 \| \alpha_2 \| \ldots \| \alpha_n$ cu $|\alpha_i| = p$ pentru $i < n$, $\quad |\alpha_n| \leq p$. $OMAC(x)$ se defineşte după următorul algoritm:

1. Fie $L = e_K(00\ldots0)$. Se determină $H_L(C_1)$ şi $H_L(C_2)$.
   (Acest pas poate fi preprocesat pentru o cheie dată K, el nedepinzând de mesajul x).
2. Dacă $|\alpha_n| < p$, α_n se completează cu bitul $'1'$ urmat – eventual – de un şir arbitrar de biţi, până ajunge la lungimea p. Spunem că α_n "a fost completat".
3. $\alpha_n := \begin{cases} \alpha_n \oplus H_L(C_1) & \text{dacă } \alpha_n \text{ nu a fost completat} \\ \alpha_n \oplus H_L(C_2) & \text{dacă } \alpha_n \text{ a fost completat} \end{cases}$
4. Se determină CBC MAC-ul pentru $\alpha_1 \| \alpha_2 \| \ldots \| \alpha_n$.
5. $OMAC(x)$ constă din primii t biţi din acest MAC.

Versiunea (curentă) $OMAC1$ este implementată în felul următor:
Se consideră $\mathcal{P} = \mathcal{C} = \{0,1\}^p$, unde $p = 64$ sau $p = 128$. Toate toate operaţiile se efectuează în $GF(2^p)$, inel structurat astfel: fiecare $\alpha = a_1 a_2 \ldots a_p \in Z_2^p$ este asociat cu polinomul $\alpha(X) = a_1 X^{p-1} + \ldots + a_{p-1} X + a_p \in GF(2^p)$. Operaţiile pe $GF(2^p)$ sunt definite:
- adunarea $\alpha + \beta = \gamma$ este un XOR bit-cu-bit: dacă $\alpha = a_1 \ldots a_p$, $\beta = b_1 \ldots b_p$, $\gamma = c_1 \ldots c_p$, atunci $c_i = a_i \oplus b_i$ $(1 \leq i \leq p)$.
- înmulţirea $\alpha' = \alpha \cdot X$ se face după algoritmul

1. Se ia vectorul α.
2. Se elimină primul bit (din stânga) şi se inserează la sfârşit bitul $'0'$; fie β vectorul obţinut.
3. Dacă bitul eliminat a fost $'1'$, atunci $\alpha' = \begin{cases} \beta \oplus 00\ldots1B & \text{dacă } p = 64 \\ \beta \oplus 00\ldots87 & \text{dacă } p = 128 \end{cases}$

   altfel $\alpha' = \beta$.

Funcţia $H_L(x)$ se defineşte $H_L(x) = L \cdot x$, iar constantele $C_1 = 00\ldots2,\ C_2 = 00\ldots4$ (în $GF(2^p)$, C_1 corespunde polinomului X, iar C_2 – polinomului X^2). Deci

$$H_L(C_1) = L \cdot X, \quad H_L(C_2) = H_L(C_1) \cdot X.$$

2.2.3 Modul autentificat de operare

După cum am văzut, un MAC asigură autentificarea şi integritatea unui mesaj. Nu s-a pus problema confidenţialităţii, mesajul în discuţie fiind un text clar.
Uneori însă este nevoie ca mesajele criptate să fie protejate prin algoritmi de autentificare şi integritate. Acest lucru este realizat de obicei combinând operaţia de criptare cu cea de construire a unui MAC; procedeul este numit *Mod autentificat de operare* (Authenticated Mode of Operation).

Sunt cunoscute diverse astfel de protocoale. Unul dintre cele mai frecvent folosite în acest moment este numit CCM (Counter with CBC-MAC) şi este o combinaţie cu AES (sau alt sistem simetric de criptare pe blocuri de 128 biţi).

Mai exact, fiind dat un mesaj α, se determină $T = MAC(\alpha)$ şi apoi se criptează $T\|\alpha$.
Pentru construcţia unui CCM folosesc:

- O cheie K a sistemului de criptare bloc pe 128 biţi.

- Doi parametri:
  M – mărimea (în octeţi) a lui T; M este un număr par în intervalul $[4, 16]$;
  L – mărimea (în octeţi) a zonei care codifică lungimea mesajului α; L este un număr în intervalul $[2, 8]$.
  M şi L sunt codificate pe câte 3 biţi fiecare, anume reprezentarea în binar a lui $(M-2)/2$ respectiv $L-1$. Valoarea $L = 1$ este rezervată (pentru aplicaţii viitoare, când lungimea mesajelor va depăşi 256^8 octeţi – cam 2^{34} GB).

- Un nonce N de $15 - L$ octeţi.

- O valoare a de autentificare suplimentară (de exemplu un număr într-o secvenţă dintr-o sesiune de lucru, sau headerul unui pachet de date).

- $flag_1$: un octet definit
  $$flag_1 = 0\|adata\|M\|L$$
  unde $adata$ este un bit setat pe $'0'$ dacă şi numai dacă $|a| = 0$.

În prima etapă se calculează un CBC MAC pentru α în felul următor:

1. Se determină blocul (de 128 biţi)
$$B_0 = flag_1 \| N \| |\alpha|$$
2. Se descompune α în blocuri de 128 biţi: $\alpha = B_1 \| B_2 \| \ldots \| B_n$
(eventual ultimul bloc se completează cu zero la sfârşit).
3. Dacă $adata = 1$, se construiesc câteva blocuri $B_0^1, \ldots B_0^r$ formate din $|a|$ urmat de a, completate apoi cu 0 până la un multiplu de 128 biţi.
4. Se calculează CBC MAC-ul mesajului $B_0 \| B_0^1 \| \ldots \| B_0^r \| B_1 \| \ldots \| B_n$.
5. Se reţin în T primii M octeţi ai rezultatului.

În etapa a doua (de criptare) se procedează după algoritmul următor:

1. Se construiesc blocurile (de numărare) $A_0, A_1, \ldots$ definite
$$A_i = flag_2 \| N \| i$$
unde numărul i este codificat pe L octeţi, iar $flag_2$ este un octet cu L pe primii 3 biţi şi 0 în rest.
Numărul acestor blocuri depinde de $|\alpha|$.
2. T se criptează în maniera sistemelor fluide de criptare (se face un XOR între T şi primii M octeţi din $e_K(A_0)$; a se vedea [2], pag. 47-59).
3. α se criptează prin XOR-are cu primii $|\alpha|$ octeţi din $e_K(A_1) \| e_K(A_2) \| \ldots$
4. Mesajul final este concatenarea celor două texte criptate obţinute anterior.

2.3 Canale subliminale

Noţiunea de *canal subliminal* a fost introdusă de Simmons în 1984 ([74]); acesta a plecat în consideraţiile sale de la **problema prizonierilor**:

Doi complici la o crimă au fost arestaţi şi închişi în celule separate. Singura lor modalitate de comunicare este folosirea de mesaje scrise şi transmise prin curieri, despre care toată lumea ştie că sunt agenţi ai gardienilor.

Gardienii permit schimbul de mesaje atât pentru a avea controlul comunicaţiilor, cât şi în speranţa că pot păcăli pe cel puţin unul din deţinuţi să accepte un mesaj conceput (sau cel puţin unul modificat) de ei. În plus, deoarece există suspiciunea că deţinuţii

plănuiesc o evadare, toate mesajele sunt citite şi se transmit numai dacă conţinutul lor este considerat inocent.

Cei doi deţinuţi sunt conştienţi de aceste riscuri. Ei trebuie însă să accepte comunicarea în orice condiţii, pentru a-şi putea coordona planul de evadare. Singura lor posibilitate este de a înşela gardienii, stabilind un mod secret de comunicare în cadrul mesajelor aflate sub control.
Deoarece ştiu că se pot introduce mesaje false sau se pot modifica mesaje, arestaţii vor lua în considerare numai mesajele pe care le pot autentifica.

Canalul subliminal este deci un canal ascuns (*cover channel*) care permite transmiterea de mesaje destinate numai celor autorizaţi.

Exemplul 2.3. *Să presupunem că deţinuţii cad de acord ca textele transmise să ascundă mesaje în modul următor: Din fiecare frază se păstrează doar cuvintele aflate pe poziţii impare. Din acestea, fiecare cuvânt de lungime pară reprezintă bitul $'0'$, iar fiecare cuvânt de lungime impară – bitul $'1'$. În acest mod, propoziţia*

"De treci codrii de aramă, de departe vezi albind, Şi auzi mândra glăsuire a pădurii de argint"

reprezintă mesajul 001100010.

Exemplul 2.4. *Să considerăm un canal ascuns elementar, care asigură transmiterea mesajelor de un bit. Atât deţinuţii cât şi gardienii sunt de acord cu transmiterea de criptograme de trei caractere binare. Mulţimea textelor criptate posibile poate fi partiţionată în două submulţimi:*

$$C_0 = \{000, 011, 101, 110\}, \qquad C_1 = \{111, 010, 100, 001\}$$

Deci în C_0 se află secvenţele de pondere pară, iar în C_1 – cele de pondere impară. Orice secvenţă din C_i reprezintă criptarea bitului i $(i = 0, 1)$. Acest lucru îl cunosc atât deţinuţii cât şi gardienii.

Deţinuţii cad de acord în prealabil asupra a două secvenţe particulare (câte una din fiecare mulţime) care să reprezinte caracterul ales de ei. Această convenţie constituie cheia secretă.

De exemplu, dacă sunt alese 011 *pentru mesajul* 0 *respectiv* 010 *pentru* 1*, în orice mesaj trimis de deţinuţi, doar aceste două secvenţe vor avea valoare pentru ei; celelalte sunt fără semnificaţie.*

Astfel, textul criptat

$$100, \quad 010, \quad 000, \quad 011, \quad 101$$

va reprezenta pentru gardieni 11000*, iar pentru deţinuţi – care păstrează doar al doilea şi al patrulea triplet –* 10*.*

O încercare a gardienilor de a trimite un mesaj fals are probabilitate de reuşită de 25%*. La fel, încercarea de modificare a mesajului (adică alegerea unui triplet din mulţimea complementară) are aceeaşi probabilitate de succes.*

De remarcat că un astfel de tip de comunicare asigură autentificarea mesajelor, nu şi confidenţialitatea: gardienii cunosc mulţimile C_0, C_1 precum şi semnificaţiile lor.

2.3.1 Canalul subliminal Simmons

Construcţia acestui canal de comunicaţie se bazează pe problema factorizării.

Fie p, q, r numere prime mari şi $n = p \cdot q \cdot r$. Pentru un număr $a \in Z_n^*$, conform teoremei chineze a resturilor([2], pag. 225), ecuaţia

$$x^2 \equiv a \pmod{n}$$

are 0 sau 8 soluţii.

Canalul ascuns Simmons – pentru transmiterea unui bit m – este:

1. *Alice* şi *Bob* cad de acord în prealabil asupra a unei perechi de numere (i, j), $1 \leq i < j \leq 8$.
   Valoarea i este asociată lui $'0'$, iar j este asociată lui $'1'$.
2. *Alice* vrea să transmită mesajul criptat $a \in Z_n^*$, ales astfel ca congruenţa $x^2 \equiv a \pmod{n}$ să aibă 8 rădăcini.
   Ordonează crescător aceste rădăcini; fie acestea $a_1, a_2, \ldots, a_8$.
3. Dacă *Alice* intenţionează să transmită mesajul ascuns $'0'$, va trimite lui *Bob* perechea (a, a_i); dacă vrea să trimită mesajul ascuns $'1'$, va expedia (a, a_j).
4. La primirea perechii (a, b), *Bob* rezolvă sistemul

   $$x^2 \equiv a \pmod{p}, \quad x^2 \equiv a \pmod{q}, \quad x^2 \equiv a \pmod{r}$$

   şi află cele 8 soluţii ale congruenţei $x^2 \equiv a \pmod{n}$.
5. Ordonează aceste soluţii şi vede pe ce poziţie este b; dacă b este pe poziţia i, va obţine mesajul clar $m = 0$; dacă este pe poziţia j, va obţine $m = 1$.
   Oricare din celelalte valori nu vor duce la un text clar autentificat.

De remarcat că încercarea de a rezolva direct ecuaţia $x^2 \equiv a \pmod{n}$, fără a factoriza pe n, conduce la problema resturilor pătratice (problemă $\mathcal{NP}$ - completă).

Exemplul 2.5. *Fie $p = 3$, $q = 5$, $r = 7$; deci $n = 105$. Alice alege $a = 64$. Ecuaţia $x^2 \equiv 64 \pmod{105}$ este echivalentă cu sistemul de congruenţe:*

$$x^2 \equiv 64 \pmod{3} \text{ are soluţiile } 1 \text{ şi } 2;$$

$x^2 \equiv 64 \ (mod\ 5)$ *are soluţiile* 2 *şi* 3*;*
$x^2 \equiv 64 \ (mod\ 7)$ *are soluţiile* 1 *şi* 6*.*

Combinând aceste soluţii – cu teorema chineză a resturilor – se obţin cele 8 *soluţii ale ecuaţiei iniţiale (ordonate crescător):*

$$8,\ \ 13,\ \ 22,\ \ 43,\ \ 62,\ \ 83,\ \ 92,\ \ 97$$

Dacă valorile secrete alese de Alice şi Bob sunt $i = 1,\ j = 4$*, atunci mesajul* $(64, 8)$ *va însemna pentru Bob autentificarea lui* 64 *împreună cu textul clar ascuns* $m = 0$*, iar* $(64, 43)$ *– autentificarea lui* 64 *împreună cu textul clar ascuns* $m = 1$*.*

Celelalte perechi $(64, 13),\ (64, 62),\ (64, 83),\ (64, 92),\ (64, 97)$ *sunt respinse de Bob ca neautentice.*

2.3.2 Canalul subliminal Ong - Schnorr - Shamir

În această schemă, *Alice* şi *Bob* dispun de un număr p mare (eventual prim) şi o cheie secretă k, $(k, p) = 1$ – cunoscută doar de cei doi parteneri.

Dacă *Alice* doreşte să trimită lui *Bob* un mesaj subliminal $y \in Z_p$ folosind textul criptat $x \in Z_p$, $(x, p) = 1,\ (y, p) = 1$, va folosi schema următoare:

1. *Alice* calculează "autentificatorii"

$$\alpha = \frac{\frac{x}{y} + y}{2} \ (mod\ p), \qquad \beta = k \cdot \frac{\frac{x}{y} - y}{2} \ (mod\ p)$$

2. Trimite lui *Bob* tripletul (x, α, β).

3. *Bob* calculează (pentru autentificare)

$$x' = \alpha^2 - \frac{\beta^2}{k^2} \ (mod\ p)$$

Dacă $x' = x$, mesajul este autentic.

4. *Bob* află mesajul subliminal folosind formula

$$y = \frac{x}{\alpha + \frac{\beta}{k}} \ (mod\ p)$$

Exemplul 2.6. *Fie* $p = 15$ *şi* $k = 2$*; deci* $k^{-1} = 8 \ (mod\ 15)$*.*

Să presupunem că Alice doreşte să trimită lui Bob mesajul subliminal $y = 4$*, folosind textul criptat* $x = 13$*.*
La primul pas, ea calculează $\alpha = 13,\ \beta = 3$ *şi trimite apoi lui Bob tripletul* $(13, 13, 3)$*.*

La recepţie, Bob determină $x' = 13$; deoarece $x' = x$, mesajul este autentic, aşa că se trece la calculul mesajului subliminal, pe baza formulei de la pasul 4.

Dacă Bob ar fi primit tripletul $(7, 2, 4)$, la pasul 3 ar fi obţinut valoarea $x' = 8$, care este diferită de $x = 7$. Deci mesajul nu este autentic.

2.3.3 Canalul subliminal *El Gamal*

În sistemul de autentificare *El Gamal*, *Alice* alege un număr prim mare q şi un element primitiv $\alpha \in Z_q$. Valorile q şi α sunt publice.

Printr-un canal ascuns, *Alice* şi *Bob* stabilesc un număr $p \in Z_q^*$.

Să presupunem acum că *Alice* vrea să trimită lui *Bob* mesajul subliminal $y \in Z_q$, folosind textul criptat $x \in Z_q$. Protocolul dintre cele două părţi este următorul:

1. *Alice* calculează $\beta = \alpha^y \pmod q$.
2. Determină γ ca soluţie a ecuaţiei
$$x = p \cdot \beta + y \cdot \gamma \pmod{(q-1)}$$
3. Trimite lui *Bob* tripletul (x, β, γ).
4. *Bob* calculează
$$a = (\alpha^p)^\beta \cdot \beta^\gamma \pmod q$$
5. Dacă $a \equiv \alpha^x \pmod q$, atunci mesajul este autentic.
6. *Bob* află mesajul subliminal
$$y = \frac{x - p \cdot \beta}{\gamma} \pmod{(q-1)}$$

Consistenţa congruenţei de la pasul 5 se obţine prin calculul:
$$a = (\alpha^p)^\beta \cdot \beta^\gamma \pmod q = \alpha^{x - y\cdot\gamma} \cdot \alpha^{y\cdot\gamma} = \alpha^x \pmod q$$

Exemplul 2.7. *Să considerăm $q = 11$ şi $\alpha = 2$ un generator al lui Z_{11}.*
Presupunem că Alice şi Bob stabilesc drept cheie secretă $k = 5$.

Dacă Alice vrea să trimită lui Bob mesajul subliminal $y = 9$ folosind textul criptat $x = 5$, ea va determina întâi
$$\beta = \alpha^y = 2^9 = 6 \pmod{11}$$
apoi va rezolva ecuaţia
$$5 = 8 \cdot 6 + 9 \cdot \gamma \pmod{10}$$

cu soluţia $\gamma = 3$.

Deci tripletul trimis este $(5, 6, 3)$.

La recepţie, Bob calculează $a = (\alpha^p)^\beta \cdot \beta^\gamma \pmod q = (2^8)^6 \cdot 6^3 = 10 \pmod{11}$

şi $\alpha^x = 2^6 = 10 \pmod{11}$.

Cum cele două valori sunt egale, Bob decide că mesajul este autentic, şi trece la aflarea mesajului subliminal:

- determină întâi $3^{-1} = 7 \pmod{10}$*, apoi*

- $y = \dfrac{x - p \cdot \beta}{\gamma} = \dfrac{5 - 8 \cdot 6}{3} = 9 \pmod{10}$.

2.3.4 Canalul subliminal Seberry - Jones

Sistemul are ca bază un algoritm mai vechi de autentificare, numit *schema de autentificare rapidă a lui Shamir*; pentru comparaţie, vom începe prin a prezenta această schemă.

Schema de autentificare Shamir

Protocolul, bazat pe problema rucsacului, este propus de Shamir în 1978. În 1984 Odlyzco construieşte un atac reuşit asupra sa ([63]).

Fie n un număr natural şi p $(q \geq 2^n)$ un număr prim.
În faza de construcţie, *Alice* efectuează următoarele operaţii:

1. Generează aleator o matrice binară $K_{n \times 2n} = (k_{ij})$ ale cărei elemente sunt secrete.
2. Determină un vector $A = (a_1, a_2, \ldots, a_{2n})$, $a_i \in Z_q$ care verifică relaţia

$$K \cdot A^T = \begin{pmatrix} 1 \\ 2 \\ \vdots \\ 2^{n-1} \end{pmatrix} \pmod q$$

   n elemente sunt generate aleator, iar celelalte n se determină din rezolvarea acestei ecuaţii matriciale.
3. Perechea (A, q) este publică.

Să presupunem acum că *Alice* vrea să trimită lui *Bob* un mesaj $m \in Z_q$. Pentru autentificarea lui, ea construieşte vectorul $C = (c_1, c_2, \ldots, c_{2n})$ în felul următor:

1. Reprezintă m în binar: $m = a_1 a_2 \dots a_n, \;\; a_i \in \{0, 1\}$.
2. Construieşte simetricul $\tilde{m} = a_n \dots a_2 a_1$.
3. Calculează $C = \tilde{m} \cdot K \pmod q$.
4. Trimite perechea (m, C).

La recepţie, *Bob* verifică autenticitatea lui m controlând dacă $C \cdot A^T = m$. Într-adevăr,

$$C \cdot A^T = (\tilde{m} \cdot K) \cdot A^T = \tilde{m} \left(K \cdot A^T \right) = \tilde{m} \cdot (1 \; 2 \; \dots \; 2^{n-1})^T = m \pmod q$$

Un factor de insecuritate îl constituie faptul că fiecare mesaj trimis conduce la construirea unui sistem de $2n$ ecuaţii liniare, din care se pot obţine unele informaţii referitoare la structura matricii de autentificare. Odlyzko a arătat că, în general, după interceptarea a n mesaje, *Oscar* poate deduce matricea K.

Ulterior, Shamir aduce unele îmbunătăţiri sistemului, adăugând la fiecare mesaj un vector binar aleator $R = (r_1, \dots, r_{2n})$.
Autentificatorul C asociat mesajului $m \in Z_q$ este construit astfel:

1. Se calculează $\alpha = m - R \cdot A^T \pmod q$.
2. Se determină $C' = \tilde{\alpha} \cdot K \pmod q$ şi apoi $C = C' + R$.
3. Se trimite prin canal perechea (C, m).

Relaţia de verificare a autenticităţii este aceeaşi cu cea din schema iniţială:

$$C \cdot A^T = (C' + R) \cdot A^T = C' \cdot A^T + R \cdot A^T = \tilde{\alpha} \cdot K \cdot A^T + R \cdot A^T = \alpha + R \cdot A^T = m \pmod q$$

Exemplul 2.8. *Fie $n = 3$, $q = 7$ şi matricea secretă*

$$K = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Pentru vectorul A, alegem aleator (din Z_7) primele trei componente; să presupunem că

$a_1 = 1,\ a_2 = 3,\ a_3 = 4$. *Restul componentelor se determină rezolvând ecuaţia matricială*

$$\begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 3 \\ 4 \\ a_4 \\ a_5 \\ a_6 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 4 \end{pmatrix} \qquad (mod\ 7)$$

Se obţine în final $A = (1\ 3\ 4\ 4\ 1\ 2)$.

Să tratăm cele două variante ale schemei rapide Shamir.

1. *Nu se foloseşte vectorul aleator* R.
   Atunci un text clar – să spunem $m = 3$ *– este autentificat prin*

$$C = \tilde{m} \cdot K = (1\ 1\ 0) \cdot \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} = (1\ 1\ 2\ 1\ 1\ 0)$$

   La primirea perechii $(m, C) = (3, (1\ 1\ 2\ 1\ 1\ 0))$ *şi folosind cheia publică* (A, q), *Bob autentifică* m *calculând*

$$C \cdot A^T = (1\ 1\ 2\ 1\ 1\ 0) \cdot (1\ 3\ 4\ 4\ 1\ 2)^T = 17 \equiv 3 \pmod 7$$

2. *Se foloseşte un vector binar aleator* R; *fie acesta* $R = (0\ 1\ 1\ 1\ 0\ 1)$. *Autentificatorul pentru mesajul* $m = 3$ *se calculează treptat astfel:*
   $\alpha = m - R \cdot A^T = 3 - (0\ 1\ 1\ 1\ 0\ 1) \cdot (1\ 3\ 4\ 4\ 1\ 2)^T = 3 - 13 \equiv 4 \pmod 7$.
   $\alpha = 4$ *se converteşte în secvenţa binară* $(1\ 0\ 0)$ *şi se inversează:* $\tilde{\alpha} = (0\ 0\ 1)$.

$$C' = \tilde{\alpha} \cdot K = (0\ 0\ 1) \cdot \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} = (1\ 0\ 0\ 0\ 1\ 1).$$

   $C = C' + R = (1\ 0\ 0\ 0\ 1\ 1) + (0\ 1\ 1\ 1\ 0\ 1) = (1\ 1\ 1\ 1\ 1\ 2)$.

   La primirea perechii $(3, (1\ 1\ 1\ 1\ 1\ 2))$, *Bob verifică autenticitatea mesajului* m:

$$C \cdot A^T = (1\ 1\ 1\ 1\ 1\ 2) \cdot (1\ 3\ 4\ 4\ 1\ 2)^T = 17 \equiv 3 \pmod 7$$

De remarcat că generarea autentificatorului C se realizează folosind reprezentarea binară a mesajului m, în timp ce procesul de autentficare operează în Z_q.

Canalul Seberry Jones

Fie q un număr prim mare şi $n = \lceil log_2 q \rceil$. Toate operaţiile vor avea loc în corpul Z_q.
În faza de pregărire a canalului subliminal, *Alice* construieşte următoarele elemente:

1. O matrice $K \in \mathcal{M}_{n\times 2n}(Z_q)$;
2. Un vector public $A = (a_1, \ldots, a_{2n})$, $a_i \in Z_q$. n componente ale lui A sunt generate aleator, iar celelalte sunt determinate de ecuaţia matricială
$$K \cdot A^T = \begin{pmatrix} 1 \\ 2 \\ \vdots \\ 2^{n-1} \end{pmatrix}$$
3. Un vector secret $B = (b_1, \ldots, b_{2n})$, $b_i \in Z_q$ cu $K \cdot B^T = 0$.
   Vectorul B este trimis lui *Bob* printr-un canal ascuns.

Să presupunem acum că *Alice* vrea să trimită lui *Bob* mesajul subliminal $y \in Z_q$, folosind mesajul $x \in Z_q$. Pentru aceasta:

1. Determină un vector binar $R = (r_1, r_2, \ldots, r_{2n})$ din ecuaţia
$$Y = R \cdot B^T \pmod q.$$
2. Criptează mesajul x în modul următor:
   (a) $x' = x - R \cdot A^T \pmod q$.
   (b) $\alpha' = x'_{(2)} \cdot K$, unde $x'_{(2)}$ este reprezentarea în binar a lui x'.
   (c) $\alpha = \alpha' + R$ (de remarcat că α şi α' sunt vectori cu $2n$ componente).
3. Trimite lui *Bob* perechea (x, α).

La recepţie, *Bob*:

1. Verifică egalitatea $\alpha \cdot A^T = x; \pmod q$. Dacă este îndeplinită, atunci mesajul x este autentic.
2. Determină mesajul subliminal y prin
$$\alpha \cdot B^T = (\alpha' + R) \cdot B^T = \left(x'_{(2)} \cdot K + R\right) \cdot B^T = x'_{(2)} \cdot K \cdot B^T + R \cdot B^T = R \cdot B^T = y \pmod q$$

Exemplul 2.9. *Să presupunem că Alice şi Bob vor să trimită mesaje subliminale din* Z_5. *Deci* $q = 5$ *şi* $n = \lceil log_2 5 \rceil = 3$.

Alice alege aleator matricea K fie aceasta

$$K = \begin{pmatrix} 2 & 2 & 4 & 1 & 3 & 1 \\ 4 & 4 & 1 & 1 & 0 & 2 \\ 4 & 3 & 4 & 2 & 2 & 0 \end{pmatrix}$$

Similar schemei rapide Shamir (Exemplul 2.8), se determină un vector A din

$$K \cdot A^T = \begin{pmatrix} 2 & 2 & 4 & 1 & 3 & 1 \\ 4 & 4 & 1 & 1 & 0 & 2 \\ 4 & 3 & 4 & 2 & 2 & 0 \end{pmatrix} \cdot (2\ 2\ 2\ a_4\ a_5\ a_6)^T = \begin{pmatrix} 1 \\ 2 \\ 4 \end{pmatrix} \quad (mod\ 5)$$

Se găseşte $A = (2\ 2\ 2\ 2\ 4\ 1)$. Vectorul B se determină din

$$K \cdot B^T = \begin{pmatrix} 2 & 2 & 4 & 1 & 3 & 1 \\ 4 & 4 & 1 & 1 & 0 & 2 \\ 4 & 3 & 4 & 2 & 2 & 0 \end{pmatrix} \cdot (b_1\ b_2; b_3\ b_4\ b_5\ b_6)^T = 0 \quad (mod\ 5)$$

Deoarece Alcie are la dispoziţie 3 ecuaţii şi 6 necunoscute, va fixa trei din ele (cu valori din Z_5); fie acestea $b_1 = 1,\ b_2 = 2,\ b_3 = 4$.
Atunci elementele rămase vor fi $b_4 = 4,\ b_5 = 3,\ b_6 = 0$.
Deci cheia secretă este $B = (1\ 2\ 4\ 4\ 3\ 5)$.

Dacă acum Alice doreşte să trimită mesajul subliminal $y = 4$ odată cu textul clar $x = 3$, ea va calcula întâi vectorul binar R din ecuaţia

$$(r_1\ r_2\ r_3\ r_4\ r_5\ r_6) \cdot (1\ 2\ 4\ 4\ 3\ 0)^T = 4 \quad (mod\ 5)$$

Sunt multe secvenţe binare care verifică această ecuaţie. Să presupunem că a fost aleasă soluţia $R = (1\ 0\ 1\ 1\ 0\ 0)$. Mai departe, Alice calculează secvenţa de autentificare:
$x' = x - R \cdot A^T = 3 - (1\ 0\ 1\ 1\ 0\ 0) \cdot (2\ 2\ 2\ 2\ 4\ 1)^T = 2\ \ (mod\ 5)$.

$$\alpha' = x'_{(2)} \cdot K = (0\ 1\ 0) \cdot \begin{pmatrix} 2 & 2 & 4 & 1 & 3 & 1 \\ 4 & 4 & 1 & 1 & 0 & 2 \\ 4 & 3 & 4 & 2 & 2 & 0 \end{pmatrix} = (4\ 4\ 1\ 1\ 0\ 2).$$

$\alpha = \alpha' + R = (4\ 4\ 1\ 1\ 0\ 2) + (1\ 0\ 1\ 1\ 0\ 0) = (0\ 4\ 2\ 2\ 0\ 2)$.

Perechea $(3, (0\ 4\ 2\ 2\ 0\ 2))$ este trimisă lui Bob.

Acesta efectuează înmulţirea

$$\alpha \cdot A^T = (0\ 4\ 2\ 2\ 0\ 2) \cdot (1\ 2\ 4\ 4\ 3\ 0)^T = 3 \quad (mod\ 5)$$

Deoarece $x = 3$ este egal cu $\alpha \cdot A^T$, autentificarea este verificată.
Se recrează acum mesajul subliminal

$$y = \alpha \cdot B^T = (0\ 4\ 2\ 2\ 0\ 2) \cdot (1\ 2\ 4\ 4\ 3\ 0)^T = 4 \quad (mod\ 5)$$

2.4 Exerciţii

2.1. *Să construim un MAC folosind modul CFB de implementare, în loc de modul CBC ([2]): fiind date blocurile de text clar $\alpha_1, \ldots, \alpha_n$, definim vectorul de iniţializare $\beta_0 = \alpha_1$. Apoi criptăm secvenţa de blocuri $\alpha_2, \ldots, \alpha_n$ după formulele*

$$\beta_i = \alpha_{i+1} \oplus e_K(\beta_{i-1}) \qquad 1 \leq i \leq n-1$$

În final, $MAC(\alpha_1 \| \ldots \| \alpha_n) = e_K(\beta_{n-1})$. Arătaţi că acesta este identic cu CBC MAC.

2.2. *Fie $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ un sistem de criptare simetric cu $\mathcal{P} = \mathcal{C} = \{0,1\}^m$ şi o funcţie de dispersie $h : (\{0,1\}^m)^n \longrightarrow \{0,1\}^m$ definită (pentru cheia $K \in \mathcal{K}$):*

$$h_K(y_1, \ldots, y_n) = e_K(y_1) \oplus \ldots \oplus e_K(y_n)$$

Arătaţi că codul HMAC construit pe baza acestei funcţii nu este sigur.

2.3. *Alice şi Bob intenţionează să comunice folosind canalul subliminal Simmons. Ei convin asupra numerelor prime $p = 5,\ q = 7,\ r = 11$ şi asupra unei strategii comune de primire: anume prima rădăcină (în ordonare crescătoare) va reprezenta mesajul elementar $'0'$, iar a cincea – mesajul elementar $'1'$.*

Să se determine autentificatorii ambelor mesaje binare, la primirea mesajului criptat $c = 256$.

2.4. *Fie canalul subliminal Ong - Schnorr - Shamir dat de $n = 21$ şi cheia secretă $k = 5$. Să se determine dacă tripletul*

$$(c_1, c_2, c_3) \in \{(14, 12, 11), (11, 1, 5), (8, 18, 5)\}$$

transmite mesaje ascunse.
În caz afirmativ, să se determine aceste mesaje.

2.5. *Se consideră canalul subliminal El Gamal dat de $q = 13,\ \alpha = 6$. Să presupunem că Alice trimite mesajul $y = 9$ folosind cheia secretă $r = 10$ şi textul criptat $x = 11$.*
Să se determine parametrii (u, v).

2.6. *Folosind aceeaşi parametri din problema anterioară, să se determine dacă tripletul $(c, x, y) = (11, 5, 1)$ asigură un canal subliminal.*
În caz afirmativ, să se determine textul clar corespunzător.

2.7. *Schema El Gamal este definită peste corpul Z_q generat de elementul primitiv α. Ce se întâmplă cu procesele de criptare/decriptare dacă $\alpha \in Z_q$ nu este primitiv ?*

2.8. *Să se construiască un canal Sebery - Jones pentru transmiterea mesajelor ascunse din Z_7. Este posibilă folosirea matricii K şi a vectorului A din Exemplul 2.9 ?*
În caz afirmativ, urmăriţi procedeele de criptare şi decriptare pentru transmiterea mesajului ascuns $m = 3$.

Capitolul 3

Semnături electronice

3.1 Consideraţii generale

În acest capitol vom aborda noţiunea de semnătură electronică (într-un mediu de calcul), împreună cu unele standarde de implementare.

Orice semnătură pe un document autentifică acel document dar şi angajează – în mod normal – responsabilitatea semnatarului. Probleme practice legate de rapiditatea transmiterii unor documente care să fie certificate ca autentice prin semnătură au condus la necesitatea creerii de semnături electronice.

De exemplu, se ştie că majoritatea operaţiunilor şi tranzacţiilor bancare devin legal valide *numai după* ce ambele părţi au semnat formularele respective. Totuşi, dacă părţile sunt legate într-o reţea de calculatoare, ele vor adesea să faciliteze această operaţie care provoacă un mare consum de timp; solicită de aceea posibilitatea de a semna documentele folosind terminalele şi reţeaua aflată la dispoziţie.

Deci, apare următoarea problemă:

Cum se poate crea o semnătură într-un mediu de calcul ?

Deoarece calculatoarele acceptă informaţia numai în formă digitală, orice semnătură pusă în discuţie trebuie să aibă această formă.
O semnătură (electronică sau olografă) trebuie să satisfacă următoarele condiţii:

- **Unică**: o anumită semnătură trebuie să poată fi generată numai de o singură persoană;
- **Neimitabilă**: nici o altă persoană nu va putea genera semnătura utilizatorului indicat; altfel spus, utilizatorii ilegali trebuie să rezolve probleme $\mathcal{NP} - complete$ dacă vor să folosească o semnătură care nu le aparţine;
- **Uşor de autentificat**: orice destinatar legal şi orice arbitru (în cazul unor eventuale dispute) să poată stabili autenticitatea semnăturii (indiferent după ce interval de timp);

- **Imposibil de negat**: nici un utilizator legal să nu-şi poată nega propria semnătură, sub afirmaţia că nu este autentică;
- **Uşor de generat**.

Trebuie făcută totuşi distincţie între semnătura olografă şi cea digitală (electronică). Iată câteva diferenţe notabile între cele două tipuri de semnături:

- O semnătură scrisă de mână este o confirmare fizică a unui document, cu ajutorul unei foi de hârtie care conţine două elemente: un mesaj (textul documentului) şi o semnătură. O astfel de legătură între mesaje şi semnături nu este posibilă într-un mediu de calcul;
- O semnătură olografă este aceeaşi indiferent de document. Pentru semnăturile digitale însă, este esenţial ca ele să depindă atât de semnatar cât şi de conţinutul documentului;
- Orice copie a unui document electronic (inclusiv semnătura) este identică cu originalul. În schimb copia unui document pe hârtie este diferită ca valoare de original. Aceasta conduce la ideea că un document electronic nu este reutilizabil. De exemplu, dacă *Bob* trimite lui *Alice* un cec în valoare de 10 milioane lei, banca nu va accepta onorarea sa decât o singură dată.

3.2 Protocoale de semnătură

Orice protocol de semnătură electronică este format din două componente: un algoritm de semnătură şi un algoritm de verificare. *Alice* semnează un mesaj x bazat pe un algoritm (secret) de semnătură sig. Rezultatul $sig(x)$ este apoi verificat de un algoritm public de verificare ver. Pentru orice pereche (x, y), algoritmul de verificare oferă un răspuns dicotomic (*adevărat* sau *fals*), după cum y este o semnătură autentică a lui x sau nu. Formal ([78]):

Definiţia 3.1. *Un* **protocol de semnătură** *este un cvintuplu* $(\mathcal{P}, \mathcal{A}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ *unde:*

1. $\mathcal{P}, \mathcal{A}, \mathcal{K}$ *sunt mulţimi finite, nevide, ale căror elemente se numesc* **mesaje, semnături** *şi respectiv* **chei***;*
2. *Există o aplicaţie biunivocă între* $\mathcal{K}$*şi* $\mathcal{S} \times \mathcal{V}$*; anume, pentru fiecare* $K \in \mathcal{K}$ *există o pereche unică* (sig_K, ver_K) *unde* $sig_K : \mathcal{P} \longrightarrow \mathcal{A}$, $ver_K : \mathcal{P} \times \mathcal{A} \longrightarrow \{T, F\}$ *au proprietatea:*

$$\forall x \in \mathcal{P}, \forall y \in \mathcal{A} \ [ver_K(x, y) = T \iff y = sig_K(x)]$$

O pereche (x, y) cu $x \in \mathcal{P},\ y \in \mathcal{A},\ y = sig_K(x)$, se numeşte *mesaj semnat.*

Pentru fiecare $K \in \mathcal{K}$, funcţia sig_K şi predicatul ver_K trebuie să fie calculabile în timp polinomial; predicatul ver_K este public iar funcţia sig_K este secretă.
Pentru *Oscar*, imitarea unei semnături a lui *Alice* pentru un mesaj x trebuie să fie imposibilă (din punct de vedere al complexităţii calculului). Altfel spus, pentru un x dat, numai *Alice* este capabil să calculeze o semnătură y astfel ca $ver(x, y) = T$.

Exemplul 3.1. *Un prim exemplu de semnătură este derivat din sistemul de criptare RSA.*

Fie p, q numere prime mari şi $n = pq$. Se definesc

$$\mathcal{P} = \mathcal{A} = Z_n, \quad \mathcal{K} = \{(n, p, q, a, b) \mid n = pq,\ p, q \text{ prime}, ab \equiv 1\ (mod\ \phi(n))\}.$$

Numerele n şi b sunt publice iar p, q, a sunt secrete.
Pentru $K = (n, p, q, a, b)$ se definesc:

$$sig_K(x) = x^a\ (mod\ n)$$

$$ver_K(x, y) = T \quad \Longleftrightarrow \quad x \equiv y^b\ (mod\ n)$$

Aici Alice semnează un mesaj folosind cheia sa de decriptare din sistemul de criptare RSA; ea este singura capabilă să genereze o semnătură corectă deoarece $d_K = sig_K$ este secretă. Funcţia de verificare utilizează cheia publică de criptare RSA – e_K; oricine o poate verifica.

De remarcat că oricine poate genera o semnătură a lui *Alice* pentru un mesaj aleator x; *Oscar* poate alege un y şi calculează $x = e_k(y)$; atunci $y = sig_K(x)$.

Acest lucru poate fi prevenit folosind mesaje x cu suficient de multă redondanţă (cu anumită semnificaţie). O altă modalitate de a evita acest atac este folosirea unor funcţii de dispersie.

În [57] există o clasificare a protocoalelor de semnătură electronică în

- Semnături cu appendix;
- Semnături cu acoperirea mesajului.

Definiţia 3.1 se referă la protocoalele de semnătură cu appendix: protocolul de verificare necesită atât semnătura cât şi mesajul asociat. Aproape toate protocoalele de semnătură prezentate în caest capitol sunt de acest tip.

Protocoalele de semnătură cu acoperirea mesajului sunt mai rare şi sunt uzual folosite pentru mesaje de lungime mică. În cazul lor mesajul x este inclus în semnătură, iar apoi este recompus în cadrul protocolului de verificare.

Să vedem cum pot fi combinate procedeele de semnătură şi criptare. Presupunem că *Alice* doreşte să trimită lui *Bob* un mesaj criptat şi semnat. Pentru un text clar x dat, *Alice* determină semnătura $y = sig_{Alice}(x)$, după care criptează x şi y folosind cheia publică a lui *Bob* : $z = e_{Bob}((x, y))$.

Textul criptat z este transmis lui *Bob*. Acesta foloseşte cheia sa secretă d_{Bob} şi obţine (x, y). După aceasta, verifică semnătura lui *Alice* cu ajutorul cheii publice $ver_{Alice}(x, y)$.

Ce se întâmplă dacă Alice criptează înainte de a semna ?
Ea va calcula $z = e_{Bob}(x),\ y = sig_{Alice}(e_{Bob}(x))$ şi va trimite lui *Bob* mesajul (z, y). Acesta decriptează z, obţine x şi verifică y ca semnătură a lui z.

Pericolul constă în faptul că *Oscar* poate intercepta (z, y), înlocuieşte y cu propria sa semnătură y' şi transmite lui *Bob* mesajul (z, y').

Din acest motiv se recomandă folosirea semnăturii **înainte** de criptare.

3.3 Securitatea protocoalelor de semnătură

Să trecem în revistă principalele tipuri de atac care pot fi dezvoltate în domeniul semnăturilor electronice. Multe din ele sunt similare atacurilor standard prezentate în [2] relativ de sistemele de criptare.

În general, *Oscar* poate lansa următoarele tipuri de atac:

- *Atacul bazat pe cheie*:
  Oscar cunoaşte doar cheia publică a lui *Alice*, deci doar predicatul de verificare ver_K.

- *Atacul bazat pe text clar*[1]:
  Oscar deţine o listă de mesaje semnate anterior de *Alice*:

$$(x_1, sig_K(x_1)),\ (x_2, sig_K(x_2)), \ldots$$

- *Atacul cu text clar ales*:
  Oscar îi cere lui *Alice* să semneze o listă de mesaje. Deci el alege textele $x_1, x_2, \ldots$ şi obţine semnăturile $sig_K(x_1),\ sig_K(x_2), \ldots$

În urma unui astfel de atac, criptanalistul poate obţine anumite informaţii relative la protocolul de semnătură vizat. Astfel:

1. *Oscar* poate determina cheia privată a lui *Alice*, deci deţine funcţia de semnătură sig_K. Este o spargere totală a protocolului – de acum *Oscar* poate genera o semnătură validă pe orice mesaj.

[1] În acest capitol nu are importanţă dacă mesajele care se semnează sunt criptate sau nu. Toate elementele din $\mathcal{P}$ sunt numite "texte clare".

2. *Oscar* poate genera – cu o probabilitate semnificativă – o semnătură validă pentru un mesaj care nu îi aparţine lui *Alice*. Deci $\exists x \in \mathcal{P}$ pentru care criptanalistul – deşi nu ştie cheia K – poate genera cu probabilitate semnificativ de mare un $y \in \mathcal{A}$ cu $ver_K(x, y) = T$. Este un atac selectiv.

3. *Oscar* poate genera o semnătură validă pentru cel puţin un mesaj (atac existenţial). Mai exact, el poate găsi o pereche $(x, y) \in \mathcal{P} \times \mathcal{A}$ cu $ver_K(x, y) = T$, iar x nu este un mesaj semnat anterior de *Alice*.

De remarcat că un protocol de semnătură nu este necondiţionat sigur: fixând un mesaj x, *Oscar* poate testa (dacă are suficient timp) toate semnăturile $y \in \mathcal{A}$ până găseşte una pentru care $ver_K(x, y) = T$.
Deci, dacă ar dispune de suficient timp, *Oscar* poate totdeauna să genereze semnătura lui *Alice* pentru orice mesaj x.

Exemplul 3.2. *Să considerăm protocolul de semnătură RSA din Exemplul 3.1. Dacă* $(x_1, y_1), (x_2, y_2)$ *sunt două mesaje semnate de Alice, atunci Oscar poate genera o semnătură* u *pentru mesajul* $z = x_1 \cdot x_2 \ (mod\ n)$. *Într-adevăr,* $u = y_1 \cdot y_2 \ (mod\ n)$ *verifică*

$$ver_K(z, u) = T.$$

Este un exemplu simplu de atac existenţial bazat pe text clar.

O variantă este folosirea unui atac cu text clar ales: Oscar vrea să obţină semnătura unui mesaj x. *El găseşte două numere* x_1, x_2 *astfel ca*

$$x = x_1 \cdot x_2 \quad (mod\ n)$$

Apoi cere lui Alice semnături pentru mesajele x_1 *şi* x_2, *după care le recompune după formula anterioară.*

Multe protocoale de semnătură folosesc semnarea amprentelor: fiind dat un mesaj $x \in \mathcal{P}$, *Alice* calculează amprenta sa $h(x)$ cu o funcţie de dispersie cunoscută, după care determină $y = sig_K(h(x))$. Mesajul semnat va fi (x, y).
Din acest motiv, securitatea unui protocol de semnătură este adesea strâns legată de securitatea standardelor de dispersie.

3.4 Protocolul ISO/IEC 9796

Standardul ISO/IEC 9796 ([82]) a fost prima normă internaţională care specifică forma unui protocol pentru o schemă de semnătură dată. Un standard nu este un protocol de semnătură ci un cadru de implementare a acesteia. Schema de semnătură nu este precizată în standard, dar aplicaţiile ISO/IEC 9796 utilizează de obicei protocolul *RSA* (Exemplul 3.1) sau o variantă dedusă din sistemul de criptare *Rabin* ([2], pg. 158-160).

3.4.1 Descrierea standardului de semnătură ISO/IEC 9796

Fie d, k numere întregi ($d \leq 512,\ k \geq 1024$). Presupunem că *Alice* foloseşte cheia publică e_{Alice} şi cheia privată d_{Alice}. Definim $\mathcal{P} = Z_2^d,\ \mathcal{A} = Z_2^k$.

Standardul de semnătură ISO/IEC 9796:

1. Fie $m \in \mathcal{P}$ mesajul care trebuie semnat. El se scrie sub forma $m = m_z m_{z-1} \dots m_1$, unde $m_i \in Z_2^8$ (eventual m_z se completează la stânga cu r zerouri, pentru a avea o lungime multiplu de 8).

2. Se iau cei mai din dreapta $t = \left\lceil \frac{k-1}{16} \right\rceil$ octeţi din secvenţa infinită
$$\dots, m_z, m_{z-1}, \dots, m_1, m_z, m_{z-1}, \dots m_1, m_z, m_{z-1}, \dots, m_1$$

3. Se defineşte funcţia $S : Z_2^8 \longrightarrow Z_2^8$ astfel: dacă $x = x_1 \| x_2$ (x_1 şi x_2 sunt cele două cifre hexazecimale care compun octetul x) atunci $S(x) = \pi(x_1) \| \pi(x_2)$ unde permutarea π este
$$\pi = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & A & B & C & D & E & F \\ E & 3 & 5 & 8 & 9 & 4 & 2 & F & 0 & D & B & 6 & 7 & A & C & 1 \end{pmatrix}$$
Se inserează la stânga fiecărui octet x octetul $S(x)$.
În plus $S(m_z) \longleftarrow S(m_z) \oplus r$, unde r este scris în binar pe 4 biţi.
Se obţine astfel cu mesaj cu $2t$ octeţi având cel puţin $k-1$ biţi.

4. Se iau ultimii $k-1$ biţi din acest mesaj şi:
- Se adaugă în faţă un bit 1;
- Se înlocuieşte ultimul octet $x = x_1 \| x_2$ cu $x_2 \| 6$. Se obţine mesajul m'.

5. Se semnează mesajul m' prin $z = d_{Alice}(m')$.

Schema de extragere a mesajului din cadrul standardului ISO/IEC 9796 este:

1. Se aplică protocolul de verificare: $m' = e_{Alice}(z)$.
2. Se verifică dacă m' este de lungime k şi se termină cu cifra hexazecimală 6.
3. Se recompune mesajul m astfel:
   - Se elimină primul bit (eroare dacă nu este 1);
   - Se înlocuiesc ultimii doi octeţi $y_1y_2x_1x_2$ cu $y_1y_2\pi^{-1}(y_1)x_1$;
   - Se ia z ca ca cel mai mic index cu proprietatea $x_{2z} \oplus S(x_{2z-1}) \neq 0$ (dacă nu există astfel de z, mesajul se respinge);
   - Se ia $r \longleftarrow z$ şi se verifică dacă $r \leq 7$;
   - Se extrag octeţii $x_{2z}, x_{2z-2}, \ldots, x_2$ şi se elimină primii $r-1$ biţi din octetul x_{2z} (mesajul se respinge dacă ei nu sunt toţi egali cu 0);
4. Se verifică dacă semnarea lui m duce la mesajul primit z (redondanţa).

De remarcat că acest standard este cu acoperirea mesajului: mesajul m de lungime cel mult d este încorporat în semnătura z; procesul de verificare conduce şi la recompunerea sa.

Exemplul 3.3. *([80]): Fie mesajul*

$$PAY \;\; 1,000,000. - CFH$$

cu $k = 12$. *Mesajul – de* 18 *caractere – este reprezentat în hexazecimal de*

P	A	Y		1	,	0	0	0	,	0	0	0	.	−	C	H	F
50	40	59	20	31	27	30	30	30	27	30	30	30	2e	2d	43	48	46

Avem $z = 18$ *şi să presupunem* $t = 32$ *octeţi. Deci vom genera*

$$3127303030273030 \;\; 302e2d434846 \| 5040 \;\; 5920312730303027 \;\; 3030302e2d434846$$

După ce se completează cu funcţia S, *se ajunge la secvenţa de* 8 *cuvinte*

$$\begin{array}{cccc} 83315f278e308e30 & 8e305f278e308e30 & 8e305c2era2d9843 & 904892464e509e40 \\ 4d595e2083315f27 & 8e308e308e305f27 & 8e308e308e305c2e & 5a2d984390489246 \end{array}$$

De remarcat că $r = 1$ *(*50 *este scris pe* 4 *biţi ca* 0101*). Cum* $m_{18} = 50$ *şi* $S(50) = 4e$, *acesta se transformă în* $4e \oplus 01 = 4f$. *Noua secvenţă obţinută este*

$$\begin{array}{cccc} 83315f278e308e30 & 8e305f278e308e30 & 8e305c2era2d9843 & 904892464f509e40 \\ 4d595e2083315f27 & 8e308e308e305f27 & 8e308e308e305c2e & 5a2d984390489246 \end{array}$$

După pasul 4 *se ajunge la mesajul fial* m'*:*

$$\begin{array}{cccc} 83315f278e308e30 & 8e305f278e308e30 & 8e305c2era2d9843 & 904892464f509e40 \\ 4d595e2083315f27 & 8e308e308e305f27 & 8e308e308e305c2e & 5a2d984390489266 \end{array}$$

3.4.2 Atac asupra standardului de semnătură ISO/IEC 9796

Un prim atac a fost prezentat ([22]) de J.S. Coron, D. Naccache şi J. Stern în 1999 pe o variantă uşor modificată, anume:

- Se ignoră operaţia XOR cu reprezentarea binară a lui r;
- k este divizibil cu 64, mesajul sursă are $d = k/2$ biţi şi $t = z = k/16$.

Fie deci $\alpha = m_z \dots m_2 m_1$ scrierea (pe octeţi) a mesajului sursă α. Dacă $m_1 = m_{1s} \| m_{1d}$ este ultimul octet, vom scrie $u = m_{1d} \| 6$. Secvenţa formatată este

$$((S(m_z) \oplus 1) \vee 80) \| m_z \| S(m_{z-1}) \| m_{z-1} \| \dots \| S(m_2) \| m_2 \| S(m_1) \| u$$

În varianta simplificată, această secvenţă este

$$(S(m_z) \vee 80) \| m_z \| S(m_{z-1}) \| m_{z-1} \| \dots \| S(m_2) \| m_2 \| S(m_1) \| u$$

Să presupunem că primul bit din $S(m_z)$ este 1 şi $m_1 = 66$. Atunci secvenţa formatată este

$$S(m_z) \| m_z \| S(m_{z-1}) \| m_{z-1} \| \dots \| S(m_2) \| m_2 \| S(m_1) \| m_1$$

Cum z este divizibil cu 4, putem alege un text clar de forma

$$\alpha = m_4 m_3 m_2 m_1 \| m_4 m_3 m_2 m_1 \| \dots \| m_4 m_3 m_2 m_1$$

unde $m_1 = 66$ şi m_4 este ales astfel ca $S(m_4)$ să aibă 1 ca prim bit. Secvenţa formatată pentru acest text este

$$S(m_4) \| m_4 \| S(m_3) \| m_3 \| S(m_2) \| m_2 \| 2266 \| \dots \| S(m_4) \| m_4 \| S(m_3) \| m_3 \| S(m_2) \| m_2 \| 2266$$

Să notăm

$$x = S(m_4) \| m_4 \| S(m_3) \| m_3 \| S(m_2) \| m_2 \| 2266, \qquad \Gamma = 1 + 2^{64} + 2^{128} + \dots + 2^{k-64}.$$

Dacă x este considerat ca un număr întreg pe 64 biţi, atunci secvenţa formatată este exact $x \cdot \Gamma$.
Sunt 2^{23} mesaje α_i de acest tip pentru care secvenţa formatată este $x_i \cdot \Gamma$. Probabilitatea ca toţi factorii primi ai lui α_i să fie mai mici decât 2^{16} este $2^{-7.7}$. Deci avem mai mult de 2^{15} mesaje α_i care se descompun în factori primi mai mici de 2^{16}.
Asta înseamnă că în cel mult 200 încercări se pot găsi patru mesaje $\alpha_g, \alpha_h, \alpha_i, \alpha_j$ astfel ca $\alpha_g \cdot \alpha_h = \alpha_i \cdot \alpha_j$. Deci semnătura mesajului α_g poate fi aflată pe baza altor trei semnături, lucru realizabil printr-un atac cu text clar ales.

Ulterior, acest atac a fost îmbunătăţit şi extins la un atac asupra standardului complet ISO/IEC 9796.

3.5 Protocolul de semnătură *El Gamal*

Fie p un număr prim (pentru care problema logaritmului discret în Z_p este dificilă) şi $\alpha \in Z_p^* = Z_p \setminus \{0\}$ un element primitiv. Se defineşte:

$$\mathcal{P} = Z_p^*, \quad \mathcal{A} = Z_p^* \times Z_{p-1}, \quad \mathcal{K} = \{(p, \alpha, a, \beta) \mid a \in Z_{p-1},\ \beta = \alpha^a \ (mod\ p)\}$$

β este cheia publică iar a este cheia secretă. Valorile p şi α sunt de asemenea publice. Pentru $K = (p, \alpha, a, \beta)$ şi $k \in Z_{p-1}$ (secret) se defineşte semnătura mesajului x prin:

$$sig_K(x, k) = (\gamma, \delta) \text{ unde}$$

$$\gamma = \alpha^k \ (mod\ p),\ \delta = (x - a\gamma)k^{-1} \ (mod\ (p-1)).$$

Pentru $x, \gamma \in Z_p^*,\ \delta\ \in Z_{p-1}$ se defineşte procedura de verificare

$$ver_K(x, \gamma, \delta) = T \quad \Longleftrightarrow \quad \beta^\gamma \gamma^\delta \equiv \alpha^x \ (mod\ p)$$

Dacă semnătura este corectă, verificarea autentifică semnătura, deoarece:

$$\beta^\gamma \gamma^\delta \equiv \alpha^{a\gamma} \alpha^{k\delta} \ (mod\ p) \equiv \alpha^x \ (mod\ p)$$

(s-a folosit egalitatea $a\gamma + k\delta \equiv x \ (mod\ (p-1))$).

Protocolul de semnătură *El Gamal* a fost definit în 1985 ([36]). Ca o particularitate, el nu este determinist: pentru un mesaj dat pot exista mai multe semnături valide. Funcţia de verificare va trebui deci să accepte ca autentice toate aceste semnături.

Alice calculează semnătura folosind cheia sa secretă a şi o valoare aleatoare secretă k (generată numai pentru semnarea mesajului x).
Verificarea se realizează cu ajutorul cheii publice.

Exemplul 3.4. *: Să luăm* $p = 467,\ \alpha = 2,\ a = 127$. *Avem*

$$\beta = \alpha^a \ (mod\ p) = 2^{127} \ (mod\ 467) = 132.$$

Dacă Alice doreşte să semneze mesajul $x = 100$ *alegând valoarea* $k = 213$ *(de remarcat că* $(213, 466) = 1$ *şi* $213^{-1} \ (mod\ 466) = 431$*), va obţine*

$$\gamma = 2^{213} \ (mod\ 467) = 29 \text{ şi } \delta = (100 - 127 \cdot 29) \cdot 431 \ (mod\ 466) = 51.$$

Pentru a verifica semnătura, calculăm

$$132^{29} \cdot 29^{51} \equiv 189 \ (mod\ 467) \text{ şi } 2^{100} \equiv 189 \ (mod\ 467).$$

Semnătura este deci validă.

Să studiem securitatea protocolului de semnătură *El Gamal*.
Vom presupune că *Oscar* doreşte să falsifice semnătura pe mesajul x fără să ştie cheia secretă a.

- Dacă *Oscar* alege valoarea γ şi încearcă să găsească δ corespunzător, el va trebui să calculeze logaritmul discret $log_\gamma \alpha^x \beta^{-\gamma}$.
  Dacă strategia sa este inversă: să aleagă întâi δ şi să caute apoi γ, el va trebui să rezolve ecuaţia (de necunoscută γ):
  $$\beta^\gamma \gamma^\delta \equiv \alpha^x \pmod p.$$
  Nu se cunoaşte încă o metodă pentru rezolvarea unei astfel de probleme.

- Dacă *Oscar* alege aleator şi pe δ şi caută să obţină x, el va ajunge din nou la problema logaritmului discret, adică la calculul $log_\alpha \beta^\gamma \gamma^\delta$.

Oscar poate totuşi să semneze un mesaj aleator x în felul următor:
Fie numerele întregi i, j $(0 \le i \le p-2,\ 0 \le j \le p-2,\ (j, p-1) = 1)$.
Se efectuează calculele:

$$\gamma = \alpha^i \beta^j \pmod p; \quad \delta = -\gamma j^{-1} \pmod{p-1}; \quad x = -\gamma i j^{-1} \pmod{p-1}$$

(deoarece calculele sunt făcute modulo $(p-1)$, există j^{-1}).
(γ, δ) este o semnătură validă pentru x. Într-adevăr, se verifică

$$\beta^\gamma \gamma^\delta \equiv \beta^{\alpha^i \beta^j} (\alpha^i \beta^j)^{-\alpha^i \beta^j j^{-1}} \equiv \beta^{\alpha^i \beta^j} \alpha^{-ij^{-1}\alpha^i \beta^j} \beta^{-\alpha^i \beta^j} \equiv \alpha^{-ij^{-1}\alpha^i \beta^j} \equiv \alpha^{-\gamma i j^{-1}} \equiv \alpha^x$$

(toate calculele sunt făcute modulo p).
Să exemplificăm acest atac:

Exemplul 3.5. *Fie din nou* $p = 467,\ \alpha = 2,\ \beta = 132$.
Să presupunem că Oscar alege $i = 99,\ j = 179$ *(deci* $j^{-1} = 151 \pmod{p-1}$*);*
Oscar calculează:

$$\gamma = 2^{99} \cdot 132^{179} = 117 \pmod{467}$$
$$\delta = -117 \cdot 151 = 41 \pmod{466}$$
$$x = 99 \cdot 41 = 331 \pmod{466}$$

Deci $(117, 41)$ *este o semnătură a mesajului* 331*, ceea ce se poate verifica imediat, calculând*

$$132^{117} 117^{41} \equiv 303 \pmod{467} \text{ şi } 2^{331} \equiv 303 \pmod{467}.$$

Semnătura este deci validă.

Să arătăm o modalitate de atac prin care *Oscar* utilizează un mesaj semnat anterior de *Alice*.
Să presupunem că (γ, δ) este o semnătură validă a lui x. *Oscar* poate semna atunci alte tipuri de mesaje:

Fie h, i, j numere întregi din intervalul $[0, p-2]$ cu $(h\gamma - j\delta, p-1) = 1$.
Calculăm:

$$\lambda = \gamma^h \alpha^i \beta^j \ (mod\ p), \quad y = \delta\lambda(h\gamma - j\delta)^{-1} \ (mod\ p-1),$$

$$x' = \lambda(hx + i\delta)(h\gamma - j\delta)^{-1} \ (mod\ p-1)$$

unde $(h\gamma - j\delta)^{-1}$ este calculat modulo $(p-1)$.
Se poate verifica direct relaţia

$$\beta^\lambda \lambda^y \equiv \alpha^{x'} \ (mod\ p).$$

Deci (λ, y) este o semnătură validă a lui x'.

Aceste strategii prezentate construiesc semnături valide; se pare însă că nu este posibil ca cineva să contrafacă semnătura unui mesaj ales de *Alice*, fără să rezolve o problemă de logaritm discret.
Din acest motiv se consideră că nu există slăbiciuni în protocolul de semnătură *El Gamal*.

Să arătăm în final două maniere de a sparge acest protocol de semnătură, atunci când este aplicat neglijent.

- Dacă întregul aleator k este cunoscut, se determină imediat

  $$a = (x - k\delta)\gamma^{-1} \ mod\ (p-1)$$

  Din acest moment, *Oscar*, ştiind a, poate calcula semnăturile la fel ca *Alice*.

- Dacă se utilizează acelaşi k pentru mai multe mesaje.
  Aceasta îi permite de asemenea lui *Oscar* să determine a. El va proceda astfel:
  Fie (γ, δ_i) semnăturile mesajelor $x_i,\ i = 1, 2$. Avem:

  $$\beta^\gamma \gamma^{\delta_i} \equiv \alpha^{x_i} \ (mod\ p),\ i = 1, 2,\ \text{deci}\ \ \alpha^{x_2 - x_1} \equiv \gamma^{\delta_2 - \delta_1} \ (mod\ p)$$

  Înlocuind $\gamma = \alpha^k$ se obţine ecuaţia de necunoscută k:

  $$\alpha^{x_2 - x_1} \equiv \alpha^{k(\delta_2 - \delta_1)} \ (mod\ p),$$

  care este echivalentă cu

  $$x_2 - x_1 \equiv k(\delta_2 - \delta_1) \ (mod\ p-1).$$

  Dacă se ia $d = (\delta_2 - \delta_1, p-1),$ din $d \mid (p-1)$ şi $d \mid (\delta_2 - \delta_1)$ rezultă $d \mid (x_2 - x_1)$.
  Notând

$$x' = \frac{x_2 - x_1}{d} \qquad \delta' = \frac{\delta_2 - \delta_1}{d} \qquad p' = \frac{p-1}{d}$$

ecuaţia devine $x' \equiv k\delta'(mod\ p)$.
Cum $(\delta', p') = 1$, se poate determina $\epsilon = (\delta')^{-1}\ (mod\ p')$.

Valoarea lui k verifică deci relaţia $k \equiv x'\ (mod\ p)$, ceea ce conduce la determinarea a d candidaţi petnru valoarea lui k, daţi de relaţia $k = x' + ip'(mod\ p),\ i = 0, \ldots, d-1$.
Din aceste d valori posibile, soluţia se determină testând congruenţa $\gamma \equiv \alpha^k\ (mod\ p)$

3.6 Variante ale protocolului de semnătură *ElGamal*

În general, un mesaj este criptat şi decriptat o singură dată, fiind necesară doar securitatea sistemului de criptare. În schimb, un document semnat – cum ar fi un contract – are o valoare juridică, şi este posibil ca autenticitatea sa să fie verificată chiar şi după mai mulţi ani. Este deci important să existe criterii de securitate mai severe pentru semnătura electronică decât pentru criptare. Cum siguranţa protocolului de semnătură *ElGamal* este echivalentă cu complexitatea problemei logaritmului discret, este necesar să se folosească un modul p cât mai mare.
Un modul p de 1024 biţi conduce însă la o semnătură *ElGamal* de 2048 biţi, ceea ce o elimină din multe aplicaţii (cum ar fi exemplu smart - cardurile).

3.6.1 Algoritmul de semnătură electronică (DSA)

Fie p un număr prim de 512 biţi, q un factor prim de 160 biţi ai lui $p-1$ şi $\alpha \in Z_p^*$ o rădăcină primitivă de ordin q a unităţii. Definim

$$\mathcal{P} = Z_p^*,\ \ \mathcal{A} = Z_q \times Z_q,\ \ \mathcal{K} = \{(p, q, \alpha, a, \beta) \mid a \in Z_q^*,\ \beta \equiv \alpha^a\ (mod\ p)\}$$

β este cheia publică, iar a este cheia secretă.
Valorile p, q, α sunt de asemenea publice.
Pentru $K = (p, q, \alpha, a, \beta)$ şi pentru un număr aleator (secret) $k \in Z_q^*$ se definesc:

- $sig_K(x, k) = (\gamma, \delta)$ unde

$$\gamma = (\alpha^k\ mod\ p)\ (mod\ q) \qquad \delta = (x + a\gamma)k^{-1} \quad (mod\ q)$$

- Pentru $x \in Z_p^*,\ \gamma, \delta \in Z_q$ funcţia de verificare este definită

$$ver_K(x, (\gamma, \delta)) = T \iff (\alpha^{e_1}\beta^{e_2}\ mod\ p)\ (mod\ q)\ = \gamma$$

unde $e_1 = x\delta^{-1}\ (mod\ q), \qquad e_2 = \gamma\delta^{-1}\ (mod\ q)$

Algoritmul de semnătură electronică DSA este o variantă a protocolului de semnătură *ElGamal*, cu proprietatea că reduce substanţial lungimea semnăturii. Propus în 1991, el este inclus în 1994 în standardul DSS (Digital Signature Standard).

Diferenţe între protocoalele de semnătură *El Gamal* **şi** DSA:

1. DSA se distinge în primul rând de *El Gamal* prin faptul că asigură o semnătură de 320 biţi pe un mesaj de 160 biţi, lucrând peste corpul Z_p cu p de 512 biţi. Aceasta permite operaţii într-un corp cu circa 2^{160} elemente. Conjectura folosită este că în această bază, calculul logaritmului discret este dificil.

2. În definirea lui δ semnul $-$ s-a schimbat în $+$.
   Aceasta schimbă ecuaţia de verificare în: $\alpha^x \beta^\gamma \equiv \gamma^\delta \ (mod\ p)$.
   Dacă $(x + a\gamma, p-1) = 1$, atunci există $\delta^{-1}\ mod\ (p-1)$ şi această ecuaţie se poate scrie
   $$\alpha^{x\delta^{-1}} \beta^{\gamma\delta^{-1}} \equiv \gamma \ (mod\ p).$$

3. *Varianta Schnorr*: Să presupunem că q este un număr de 160 biţi astfel încât $q \mid (p-1)$ şi $\alpha \in Z_p^*$ este o rădăcină primitivă de ordinul q a unităţii modulo p (pentru a găsi un astfel de număr, se ia o rădăcină primitivă $\alpha_0 \in Z_p$ şi se construieşte $\alpha = \alpha_0^{(p-1)/q}\ (mod\ p)$).
   În acest fel, β şi γ sunt de asemenea rădăcini de ordinul q ale unităţii.
   Deci exponenţii lui α, β, γ se pot reduce modulo q, fără a modifica ecuaţia de verificare de mai sus.

   Protocolul lui Schnorr foloseşte şi o funcţie de dispersie criptografică
   $h : \{0,1\}^* \longrightarrow Z_q$ pe baza căreia se definesc componentele semnăturii astfel:
   $$\gamma = h(x \| \alpha^k), \quad \delta = k + a\gamma \ \ (mod\ q)$$

   Pentru $x \in \{0,1\}^*$ şi $\gamma, \delta \in Z_q$, procedura de verificare este
   $$ver_K(x, (\gamma, \delta)) = T \quad \Longleftrightarrow \quad h(x \| \alpha^\delta \beta^{-\gamma}) = \gamma$$

   Relaţia de verificare este adevărată deoarece $\alpha^\delta \beta^{-\gamma} = \alpha^k\ (mod\ p)$

Exemplul 3.6. *Fie* $q = 101,\ p = 78q + 1 = 7879$.
3 *este rădăcină primitivă în* Z_{7879}*, deci se poate lua* $\alpha = 3^{78}\ (mod\ 7879) = 170$.
Dacă alegem de exemplu $a = 75$*, obţinem* $\beta = \alpha^a\ (mod\ 7879) = 4567$.
Să presupunem că Alice doreşte să semneze mesajul $x = 1234$ *şi ia* $k = 50$ *(deci* $k^{-1}\ (mod\ 101) = 99$*); el va avea:*

$$\gamma = (170^{50}\ mod\ 7879)\ (mod\ 101) = 2518\ (mod\ 101) = 94$$

$$\delta = (1234 + 75 \cdot 94) \cdot 99\ (mod\ 101) = 97$$

Semnătura $(94, 97)$ *a mesajului* 1234 *se verifică prin calcul:*
$\delta^{-1} = 97^{-1} \ (mod\ 101) = 25,$
$e_1 = 1234 \cdot 25 \ (mod\ 101) = 45,$
$e_2 = 94 \cdot 25 \ (mod\ 101) = 27$

$$(170^{45} \cdot 4567^{27} \ mod\ 7879) \ (mod\ 101) = 2518 \ (mod\ 101) = 94$$

Semnătura este deci validă.

În varianta Schnorr trebuie calculată valoarea $h(1234\|2518)$ *unde* h *este o funcţie de dispersie, iar* 1234 *şi* 2518 *sunt reprezentate în binar. Pentru a nu intra în detalii, să presupunem* $h(1234\|2518) = 96$*. Atunci*

$$\delta = 50 + 75 \cdot 96 \ (mod\ 101) = 79$$

şi semnătura este $(96, 79)$*.*
Ea este verificată calculând

$$170^{79} \cdot 4567^{-96} \ (mod\ 7879) = 2518$$

şi verificând

$$h(1234\|2518) = 96.$$

Odată cu adoptarea ca standard DSS în decembrie 1994, protocolul DSA a suferit o singură modificare: mesajul x (din calculul lui δ şi al lui e_1) este înlocuit cu amprenta $SHA1(x)$.

Exemplul 3.7. *Să reluăm valorile lui* $p, q, \alpha, a, \beta, k$ *din Exemplul 3.6 şi să presupunem că Alice vrea să semneze amprenta* $SHA1(x) = 22$*. Ea va calcula*

$$k^{-1} \ (mod\ 101) = 50^{-1} \ (mod\ 101) = 99,$$

$$\gamma = (170^{50} \ (mod\ 7879)) \ (mod\ 101) = 2518 \ (mod\ 101) = 94$$

şi

$$\delta = (22 + 75 \times 94) \times 99 \ (mod\ 101) = 97.$$

Semnătura $(94, 97)$ *a amprentei* 22 *este verificată efectuând calculele*
$\delta^{-1} = 97^{-1} \ (mod\ 1010 = 25,$
$e_1 = 22 \cdot 25 \ (mod\ 101) = 45,$
$e_2 = 94 \cdot 25 \ (mod\ 101) = 27$
şi verificând

$$(170^{45} \cdot 4567^{27} \ (mod\ 7879)) \ (mod\ 101) = 2518 \ (mod\ 101) = 94.$$

Observaţia 3.1. *Este posibil ca Alice să obţină prin calcul* $\delta \equiv 0 \ (mod\ q)$. *Atunci ea trebuie să reia protocolul de semnătură pentru o nouă valoare aleatoare* k. *Practic, această situaţie apare însă extrem de rar: probabilitatea ca* $\delta \equiv 0 \ (mod\ q)$ *este cam* 2^{-160}.

Când DSS a fost propus ca standard în 1991, a avut mai multe critici. Astfel:

- Mulţi s-au arătat nemulţumiţi de impunerea mărimii de 512 biţi pentru modul; o mărime variabilă care să fie aleasă de beneficiari în funcţie de necesităţi ar fi fost mai convenabilă.
  Ca răspuns, $NIST$ a schimbat descrierea standardului, pentru a permite alegerea ca modul a oricărui număr divizibil cu 64, având între 512 şi 1024 biţi.
  În octombrie 2001 NIST revine şi recomandă alegerea pentru p a unui număr prim de 1024 biţi.

- Semnăturile pot fi mult mai uşor generate decât verificate. Pentru comparaţie, în sistemul RSA un exponent mic de decriptare poate conduce la un protocol de verificare mult mai rapid decât semnătura.
  Obiecţia este ridicată din considerente practice, anume:

  – Un mesaj este semnat o singură dată, dar poate fi verificat de foarte multe ori de-a lungul timpului.

  – Pe ce tipuri de calculatoare sunt efectuate aceste protocoale de semnătură şi/sau verificare ? Cea mai mare parte a aplicaţiilor folosesc calculatoare de birou, cu resurse limitate, care comunică cu sisteme puternice de calcul.

  Răspunsul dat de $NIST$ la această obiecţie a fost că este foarte puţin important ce protocol este mai simplu, având în vedere sistemele tot mai performante de calcul.

O altă variantă a protocolului DSA este protocolul de semnătură Pointcheval – Vaudenay, inclus ulterior în standardul ISO/IEC 14888. Singura modificare este înlocuirea lui $SHA1(x)$ cu $h(\gamma \| x)$, unde h este o funcţie de dispersie criptografică generală.
Cu această modificare se demonstrează că protocolul de semnătură este sigur, deoarece:

1. h devine similară unei funcţii aleatoare;

2. Problema logaritmului discret este dificilă;

3. Aplicaţia $k \mapsto (\alpha^k \ mod\ p) \ (mod\ q)$ este o funcţie neinversabilă.

3.6.2 Protocolul de semnătură $ECDSA$

În 2000 protocolul $ECDSA$ (*Elliptic Curve Digital Signature Algorithm*) a fost aprobat sub numele $FIPS\ 186-2$. Şi el este o variantă a protocolului $ElGamal$, construit pe funcţii eliptice. O formă simplificată a sa este:

Fie p un număr prim sau o putere a lui 2 şi E o curbă eliptică peste Z_p.
Fie P un punct din E de ordin q (număr prim) astfel ca problema logaritmului discret pentru P să fie dificilă. Se definesc

$$\mathcal{P} = \{0,1\}^*, \quad \mathcal{A} = Z_q^* \times Z_q^*, \quad \mathcal{K} = \{(p,q,E,P,m,Q) \mid m \in Z_{q-1},\ Q = mP\},$$

Valorile $p,\ q,\ E, P, Q$ sunt publice (Q este cheia publică), iar m este cheia secretă.
Pentru $K = (p,q,E,P,m,Q)$ şi $k \in Z_q$ ales aleator, definim

$$sig_K(x,k) = (\alpha,\beta)$$

unde

$$kP = (u,v), \quad \alpha = u\ (mod\ q), \quad \beta = k^{-1}(SHA1(x) + m\alpha) \quad (mod\ q).$$

(dacă $\alpha \cdot \beta = 0$ se alege altă valoare aleatoare pentru k).
Pentru $x \in \{0,1\}^*,\ \alpha, \beta \in Z_q^*$ verificarea este definită prin

$$w = \beta^{-1}\ (mod\ q), \qquad i = w \cdot SHA1(x)\ (mod\ q), \qquad j = w \cdot \alpha\ (mod\ q),$$

$$(u,v) = iP + jQ$$

şi

$$ver_K(x,(\alpha,\beta)) = T \quad \Longleftrightarrow \quad u\ (mod\ q) = \alpha$$

Exemplul 3.8. *([78]): Să considerăm curba eliptică $y^2 = x^3 + x + 6$ definită peste Z_{11}. Alegem parametrii protocolului de semnătură astfel:*

$$p = 11, \quad q = 13, \quad P = (2,7), \quad m = 7, \quad Q = (7,2).$$

Să presupunem că avem un mesaj x cu $SHA1(x) = 4$ şi Alice vrea să-l semneze folosind valoarea aleatoare $k = 3$. Ea va calcula

$$(u,v) = 3(2,7) = (8,3),\ \ \alpha = u\ (mod\ 13) = 8\ \text{şi}\ s = 3^{-1} \cdot (4 + 7 \cdot 8)\ (mod\ 13) = 7.$$

Deci semnătura este $(8,7)$.

Alice verifică această semnătură efectuând următoarele calcule:

$$w = 7^{-1}\ (mod\ 13) = 2, \quad i = 2 \cdot 4\ (mod\ 13) = 8, \quad j = 2 \cdot 8\ (mod\ 13) = 3,$$

Deci $(u,v) = 8P + 3Q = (8,3)$ şi $u\ (mod\ 13) = 8 = \alpha$.

Rezultă că semnătura $(8,7)$ este validă.

3.7 Protocoale de semnătură "One-time"

Ideea acestor protocoale se bazează pe faptul că funcţia care asigură semnătura este folosită pentru a semna un singur document, după care ea este abandonată (cu toate că poate fi verificată de un număr arbitrar de ori).

3.7.1 Protocolul de semnătură Lamport

Fie $k > 0$ un număr întreg şi $\mathcal{P} = \{0,1\}^k$.
Dacă $f : Y \longrightarrow Z$ este o funcţie neinversabilă, se alege $\mathcal{A} = Y^k$.
Se generează aleator $y_{i,j} \in Y$ $(1 \leq i \leq k,\ j = 0,1)$ şi fie

$$z_{i,j} = f(y_{i,j}).$$

Cheia K este lista celor $2k$ valori y (secrete) şi a celor $2k$ valori z (publice).
Pentru $K = \{(y_{i,j}, z_{i,j}) \mid 1 \leq i \leq k,\ j = 0,1\}$ se defineşte:

$$sig_K(x_1, \ldots, x_k) = (y_{1,x_1}, \ldots, y_{k,x_k})$$

şi

$$ver_K(x_1, \ldots, x_k, a_1, \ldots, a_k) = T \quad \Longleftrightarrow \quad f(a_i) = z_{i,x_i} \quad (1 \leq i \leq k).$$

Conform acestui protocol, mesajul care trebuie semnat este un şir binar de lungime k. Fiecare bit de valoare j $(j = 0,1)$ este semnat prin $z_{i,j}$, unde fiecare $z_{i,j}$ este imaginea printr-o funcţie neinversabilă a unui $y_{i,j}$.

Verificarea constă în aplicarea funcţiei f şi compararea rezultatului cu cheia publică.

Exemplul 3.9. *Fie* 7879 *un număr prim şi* $3 \in Z_{7879}$ *un element primitiv.*
Se defineşte

$$f(x) = 3^x \pmod{7879}.$$

Dacă Alice doreşte să semneze un mesaj de trei biţi, ea alege (secret) şase numere aleatoare

$$\begin{array}{lll} y_{1,0} = 5831, & y_{2,0} = 803, & y_{3,0} = 4285, \\ y_{1,1} = 735, & y_{2,1} = 2467, & y_{3,1} = 6449. \end{array}$$

Calculează apoi imaginea lor prin funcţia f:

$$\begin{array}{lll} z_{1,0} = 2009, & z_{2,0} = 4672, & z_{3,0} = 268, \\ z_{1,1} = 3810, & z_{2,1} = 4721, & z_{3,1} = 5731. \end{array}$$

Aceste numere z sunt publice.

Să presupunem că Alice vrea să semneze mesajul $x = (1,1,0)$. Semnătura ei este

$$(y_{1,1}, y_{2,1}, y_{3,0}) = (735, 2467, 4285)$$

Pentru a verifica semnătura, este suficient să se constate că:

$$3^{735} = 3810, \quad 3^{2467} = 4721, \quad 3^{4285} = 268,$$

toate calculele fiind realizate modulo 7879.

Oscar nu poate imita semnătura, deoarece f nu are inversă.
În plus, protocolul de semnătură poate fi utilizat doar pentru un singur mesaj: dacă dispune de două mesaje cu aceeaşi semnătură, *Oscar* poate imita semnătura unui nou mesaj (diferit de cele două).

De exemplu, dacă mesajele $(0, 1, 1)$ şi $(1, 0, 1)$ sunt semnate prin procedeul de sus cu $(y_{1,0}, y_{2,1}, y_{3,1})$ respectiv $(y_{1,1}, y_{2,0}, y_{3,1})$, se pot semna ulterior mesaje cum ar fi $(1, 1, 1)$ sau $(0, 0, 1)$.

3.7.2 Protocolul Bose - Chaum

Deşi foarte simplu şi elegant, protocolul Lamport nu este practic din cauza dimensiunii mari a semnăturii.
Reluând Exemplul 3.9, o implementare sigură necesită un modul p de 512 biţi.
Aceasta înseamnă că fiecare bit al mesajului are o semnătură de 512 biţi; avem deci o semnătură de 512 ori mai lungă decât mesajul !

De aceea a apărut o simplificare a protocolului de semnătură Lamport, care reduce lungimea semnăturii fără a diminua securitatea ei.
Numit **Bos - Chaum**, acest protocol este definit astfel:

Fie $k > 0$ un număr întreg şi $\mathcal{P} = \{0, 1\}^k$.
Dacă n este un număr întreg cu proprietatea $2^k \leq C_{2n}^n$, fie B mulţimea numerelor întregi din intervalul $[1, 2n]$ şi

$$\phi : \mathcal{P} \longrightarrow \mathcal{B}$$

o funcţie injectivă în mulţimea $\mathcal{B}$ a părţilor lui B de n elemente.
Dacă $f : Y \longrightarrow Z$ este o funcţie neinversabilă, fie $\mathcal{A} = Y^n$.
Se aleg aleator valorile $y_i \in Y \quad (1 \leq i \leq 2n)$ şi fie $z_i = f(y_i)$.
Cheia K este lista celor $2n$ valori y (secrete) şi a celor $2n$ valori z (publice).
Pentru $K = \{(y_i, z_i) \mid 1 \leq i \leq 2n\}$, se definesc

$$sig_K(x_1, \ldots, x_k) = \{y_j \mid j \in \phi(x_1, \ldots x_k)\}$$

şi

$$ver_K(x_1, \ldots, x_k, a_1, \ldots, a_n) = T \iff \{f(a_i) \mid 1 \leq i \leq n\} = \{z_j \mid j \in \phi(x_1, \ldots, x_k)\}$$

Avantajul protocolului **Bos - Chaum** este acela că reduce lungimea semnăturii.
De exemplu, să presupunem că vrem să semnăm un mesaj de şase biţi ($k = 6$); cum $2^6 = 64$ şi $C_8^4 = 70$, putem lua $n = 4$. Aceasta permite semnarea mesajului cu numai patru valori y (în loc de şase la protocolul **Lamport**).
De asemenea, şi cheia este mai scurtă, cu numai opt valori z (faţă de 12 la semnătura Lamport).

Protocolul de semnătură Bos - Chaum necesită o funcţie injectivă ϕ care asociază fiecărei secvenţe de k biţi $x = (x_1, \ldots, x_k)$ o submulţime de n elemente.
Un exemplu de algoritm simplu care realizează o astfel de asociere este:

1. $x \longleftarrow \sum_{i=1}^{k} x_i 2^{i-1}$;
2. $\phi(x) \longleftarrow \emptyset$; $t \longleftarrow 2n$; $e \longleftarrow n$;
3. **while** $t > 0$ **do**
   3.1. $t \longleftarrow t - 1$;
   3.2. **if** $x > C_t^e$ **then**
   $x \longleftarrow x - C_t^e$; $e \longleftarrow e - 1$; $\phi(x) \longleftarrow \phi(x) \cup \{t + 1\}$.

Dacă vrem să facem o estimare a valorii lui n din protocolul Bos - Chaum, plecăm de la inegalitatea $2^k \leq C_{2n}^n$ în care se evaluează $C_{2n}^n = \dfrac{(2n)!}{(n!)^2}$ cu formula lui Stirling, obţinându-se $2^{2n}/\sqrt{\pi n}$. După simplificări şi logaritmare în baza 2 se ajunge la relaţia

$$k \leq 2n - \frac{log_2 n\pi}{2}$$

Asimptotic, n este de ordinul lui $k/2$, deci protocolul de semnătură Bos - Chaum reduce mărimea semnăturii lui Lamport cu aproximativ 50%.

3.8 Semnături incontestabile

Semnăturile incontestabile (*Undeniable Signatures*) au fost introduse de Chaum şi Van Antwerpen în 1989 ([13]). Ele prezintă câteva proprietăţi specifice. Astfel:

- Semnătura nu poate fi validată fără aportul semnatarului *Alice*. Aceasta o protejează pe *Alice* de difuzarea fără consimţământ a unui document pe care se pretinde că l-ar fi semnat.
  Validarea se face urmând un protocol de întrebări şi răspunsuri.

- Pentru a evita ca *Alice* să-şi nege propria semnătură, există un *protocol de dezminţire* pe care *Alice* trebuie să-l urmeze pentru a dovedi că o semnătură este falsă.
  Refuzul de a folosi acest protocol este o confirmare a autenticităţii semnăturii.

Deci, un protocol de semnătură incontestabilă este format dintr-o funcţie de semnătură, un protocol de verificare şi o procedură de dezminţire.

Algoritmul **Chaum - van Antwerpen** este:

Funcţia de semnătură:
Fie $p = 2q + 1$ un număr prim cu proprietatea că q este prim şi $\alpha \in Z_p^*$ un element de ordin q.
Pentru $1 \leq a \leq q - 1$, se defineşte $\beta \equiv \alpha^a \ (mod\ p)$.
Fie $G =< \alpha >$ subgrupul ciclic de ordin q al lui Z_p, generat de α.
Se definesc

$$\mathcal{P} = \mathcal{A} = G, \quad \mathcal{K} = \{(p, \alpha, a, \beta) \mid a \in Z_q,\ \beta \equiv \alpha^a \ (mod\ p)\}.$$

Valorile p şi α sunt publice, β este cheia publică, iar a este cheia secretă.
Pentru $K = (p, \alpha, a, \beta)$ şi $x \in G$ se defineşte

$$y = sig_K(x) = x^a \quad (mod\ p)$$

Protocolul de verificare:
Pentru $x, y \in G$, protocolul de verificare se efectuează astfel:

1. *Bob* generează aleator numerele $e_1, e_2 \in Z_q^*$;
2. *Bob* calculează $c = y^{e_1}\beta^{e_2} \ (mod\ p)$ şi-l trimite lui *Alice*;
3. *Alice* calculează $d = c^{a^{-1} \ (mod\ q)} \ (mod\ p)$ şi-l trimite lui *Bob*;
4. *Bob* admite autenticitatea semnăturii y dacă şi numai dacă

$$d \equiv x^{e_1}\alpha^{e_2} \quad (mod\ p).$$

A. Să justificăm rolul lui p şi q în acest protocol. Calculele sunt efectuate în Z_p. Este necesar totuşi ca anumite calcule să fie făcute într-un subgrup al său, de ordin prim (notat cu G).
În particular este nevoie să calculăm inverse modulo q (ceea ce justifică de ce $q = card(G)$ trebuie să fie prim). Alegând $p = 2q + 1$ cu q prim, se asigură acest deziderat şi – în plus – dimensiunea lui G este maximă, lucru de dorit deoarece mesajele de semnat sunt elemente din G.

Să arătăm întâi cum se convinge *Bob* de autenticitatea semnăturilor valide. În calculul de mai jos, exponenţii sunt reduşi modulo q.

$d \equiv c^{a^{-1}} \ (mod\ p) \equiv y^{e_1 a^{-1}}\beta^{e_2 a^{-1}} \ (mod\ p)$.

Cum $\beta \equiv \alpha^a \ (mod\ p)$, avem $\beta^{a^{-1}} \equiv \alpha \ (mod\ p)$.
De asemenea, din $y = x^a \ (mod\ p)$ rezultă $y^{a^{-1}} \equiv x \ (mod\ p)$.
Se ajunge deci la $d \equiv x^{e_1}\alpha^{e_2} \ (mod\ p)$.

Exemplul 3.10. *Fie* $p = 467$.
Cum 2 *este un element primitiv în* Z_{467}*, rezultă că* $2^2 = 4$ *este un generator al lui* G*, grupul resturilor pătratice modulo* 467*. Vom lua deci* $\alpha = 4$.
Să presupunem $a = 101$*; avem* $\beta = \alpha^a \ (mod\ 467) = 449$.
Alice semnează deci mesajul $x = 119$ *cu*

$$y = 119^{101} \ (mod\ 467) = 129.$$

Să presupunem că Bob vrea să autentifice semnătura y *şi alege pentru asta*

$$e_1 = 38, \quad e_2 = 397.$$

El calculează $c = 13$*, la care Alice răspunde cu* $d = 9$*. Bob verifică atunci relaţia*

$$119^{38} \cdot 4^{397} \equiv 9 \quad (mod\ 467).$$

Semnătura este acceptată ca autentică.

B. Să arătăm acum că *Bob* nu poate accepta o semnătură falsă drept autentică decât cu o probabilitate neglijabilă.

Teorema 3.1. *Dacă* $y \not\equiv x^a \ (mod\ p)$ *atunci Bob admite pe* y *ca semnătură autentică a lui* x *cu probabilitate* $1/q$.

Demonstraţie. Se observă că orice întrebare c corespunde la exact q perechi distincte (e_1, e_2) posibile (deoarece y şi β sunt elemente ale grupului G de ordin q prim şi în definiţia lui c, fiecare e_1 determină un e_2 unic).
Când *Alice* primeşte c, ea nu ştie ce pereche (e_1, e_2) a fost folosită în calcul.
Vom arăta că dacă $y \not\equiv x^a \ (mod\ p)$, orice răspuns distinct d nu poate fi consistent decât cu o singură pereche (e_1, e_2).

Deoarece α generează G, orice element din G se scrie ca o putere (unică modulo q) a lui α. Deci

$$c = \alpha^i, \quad d = \alpha^j, \quad x = \alpha^k, \quad y = \alpha^m$$

cu $i, j, k, m \in Z_q$ şi operaţiile aritmetice efectuate modulo p.
Să considerăm sistemul:

$$c \equiv y^{e_1}\beta^{e_2} \ (mod\ p), \qquad d \equiv x^{e_1}\alpha^{e_2} \ (mod\ p).$$

El este echivalent cu

$$i \equiv me_1 + ae_2 \ (mod\ q), \qquad j \equiv ke_1 + e_2 \ (mod\ q).$$

Cum prin ipoteză $y \not\equiv x^a \ (mod\ p)$, rezultă $m \not\equiv ak \ (mod\ q)$.

Astfel, matricea sistemului modulo q admite un determinant nenul, deci sistemul are soluţie unică.
Altfel spus, pentru orice $d \in G$, nu există răspuns corect la întrebarea c decât pentru un singur cuplu (e_1, e_2).
Deci probabilitatea ca *Alice* să răspundă corect lui *Bob* în condiţiile teoremei este $1/q$. □

C. Să construim acum procedura de dezminţire. Ea urmează de două ori protocolul de verificare.

Procedura de dezminţire:

1. *Bob* generează aleator $e_1, e_2 \in Z_q^*$;
2. *Bob* calculează $c = y^{e_1}\beta^{e_2} \ (mod\ p)$ şi-l trimite lui *Alice*;
3. *Alice* calculează $d = c^{a^{-1} \ (mod\ q)} \ (mod\ p)$ şi-l trimite lui *Bob*;
4. *Bob* verifică $d \not\equiv x^{e_1}\alpha^{e_2} \ (mod\ p)$;
5. *Bob* generează aleator $f_1, f_2 \in Z_q^*$;
6. *Bob* calculează $C = y^{f_1}\beta^{f_2} \ (mod\ p)$ şi-l trimite lui *Alice*;
7. *Alice* calculează $D = C^{a^{-1} \ (mod\ q)} \ (mod\ p)$ şi-l trimite lui *Bob*;
8. *Bob* verifică $D \not\equiv x^{f_1}\alpha^{f_2} \ (mod\ p)$;
9. *Bob* admite că y este fals dacă şi numai dacă
$$(d\alpha^{-e_2})^{f_1} \equiv (D\alpha^{-f_2})^{e_1} \quad (mod\ p)$$

Paşii $1-4$ şi $5-8$ corespund protocolului de verificare. Pasul 9 este *validarea consistenţei răspunsului*, care permite lui *Bob* să determine dacă *Alice* a calculat bine răspunsurile sale conform protocolului.

Exemplul 3.11. *Să luăm parametrii din Exemplul 3.10:*
$p = 467,\ \alpha = 4,\ a = 101,\ \beta = 449.$
Fie mesajul $x = 286$ *cu semnătura (greşită)* $y = 83$.
Alice doreşte să dezmintă această semnătură.
Fie $e_1 = 45,\ e_2 = 237$ *primele valori alese de Bob.*
El calculează $c = 305$ *şi Alice răspunde cu* $d = 109$. *Bob calculează atunci*

$$286^{45} \cdot 4^{237} \ (mod\ 467) = 149.$$

Deoarece $149 \neq 109$, *Bob trece la pasul* 5 *al protocolului.*
Să presupunem că el alege acum $f_1 = 125,\ f_2 = 9$ *şi calculează* $C = 270$ *la care Alice*

răspunde cu $D = 68$. *Bob calculează acum*

$$286^{125} \cdot 4^9 \ (mod\ 467) = 25.$$

Cum $25 \neq 68$, *Bob trece la pasul* 9 *şi efectuează testul de consistentă:*

$$(109 \cdot 4^{-237})^{125} \equiv 188\ (mod\ 467), \qquad (68 \cdot 4^{-9})^{45} \equiv 188\ (mod\ 467)$$

Acum Bob este convins că semnătura nu este valabilă.

Pentru a completa analiza protocolului **Chaum-van Antwerpen**, mai trebuie demonstrate două elemente:

- *Alice* poate să-l convingă pe *Bob* să invalideze o semnătură incorectă.
- *Alice* nu poate să-l convingă pe *Bob* să invalideze o semnătură corectă, decât cu probabilitate neglijabilă.

Teorema 3.2. *Dacă* $y \not\equiv x^a\ (mod\ p)$ *şi dacă Alice şi Bob urmează corect protocolul de dezminţire, atunci*

$$(d\alpha^{-e_2})^{f_1} \equiv (D\alpha^{-f_2})^{e_1} \quad (mod\ p).$$

Demonstraţie. Utilizând faptul că

$$d \equiv c^{a^{-1}}\ (mod\ p) \text{ şi } c \equiv y^{e_1}\beta^{e_2} \quad (mod\ p),$$

avem:

$$\begin{aligned}(d\alpha^{-e_2})^{f_1} &\equiv ((y^{e_1}\beta^{e_2})^{a^{-1}}\alpha^{-e_2})^{f_1}\ (mod\ p) \equiv y^{e_1 f_1 a^{-1}}\beta^{e_2 a^{-1} f_1}\alpha^{-e_2 f_1}\ (mod\ p)\\ &\equiv y^{e_1 f_1 a^{-1}}\alpha^{e_2 f_1}\alpha^{-e_2 f_1}\ (mod\ p) \equiv y^{e_1 f_1}\ (mod\ p).\end{aligned}$$

Un calcul similar folosind $D \equiv C^{a^{-1}}\ (mod\ p)$, $C \equiv y^{f_1}\beta^{f_2}\ (mod\ p)$ şi $\beta \equiv \alpha^a\ (mod\ p)$ arată că

$$(D\alpha^{-f_2})^{e_1} \equiv y^{e_1 f_1 a^{-1}}\ (mod\ p)$$

deci testul de consistentă de la pasul 9 reuşeşte. □

Să studiem acum cazul când *Alice* încearcă să dezmintă o semnătură validă.
Atunci – evident – ea nu va urma protocolul, şi va construi elementele d şi D fără să respecte procedura.

Teorema 3.3. *Să presupunem* $y \equiv x^a\ (mod\ p)$ *şi Bob urmează procedura de dezminţire. Dacă* $d \not\equiv x^{e_1}\alpha^{e_2}\ (mod\ p)$ *şi* $D \not\equiv x^{f_1}\alpha^{f_2}\ (mod\ p)$ *atunci probabilitatea ca*

$$(d\alpha^{-e_2})^{f_1} \not\equiv (D\alpha^{-f_2})^{e_1} \quad (mod\ p)$$

este $1 - 1/q$.

Demonstraţie. Să presupunem că avem (conform ipotezei):

$$y \equiv x^a, \quad d \not\equiv x^{e_1}\alpha^{e_2}, \quad D \not\equiv x^{f_1}\alpha^{e_2}, \quad (d\alpha^{-e_2})^{f_1} \equiv (D\alpha^{-f_2})^{e_1}$$

toate calculele fiind făcute modulo p. Vom arăta că se ajunge la o contradicţie.

Testul de consistenţă (pasul 9) se rescrie

$$D \equiv d_0^{f_1}\alpha^{f_2} \quad (mod\ p)$$

unde

$$d_0 = d^{1/e_1}\alpha^{-e_2/e_1} \quad (mod\ p)$$

nu depinde decât de paşii $1-4$ ai protocolului de verificare.
Aplicând Teorema 3.1 se obţine că y este o semnătură validă a lui d_0 cu probabilitate $1-1/q$.
Dar, prin ipoteză, y este o semnătură validă a lui x.
Deci, cu mare probabilitate vom avea $x^a \equiv d_0^a\ (mod\ p)$ adică $x = d_0$.

Pe de-altă parte, din $d \not\equiv x^{e_1}\alpha^{e_2}\ (mod\ p)$ rezultă

$$x \not\equiv d^{1/e_1}\alpha^{-e_2/e_1} \quad (mod\ p),$$

adică tocmai $x \neq d_0$, contradicţie. □

3.9 Protocoale de semnătură fără eşec

O semnătură fără eşec (*Fail - Stop Signature*) oferă protecţie contra unui adversar atât de puternic încât poate contraface semnături.

3.9.1 Protocolul de semnătură *Heijst - Pedersen*

Protocolul propus de *Heijst şi Pedersen* în 1992 ([40]) este o semnătură *One - Time*, compusă dintr-o funcţie de semnătură, o funcţie de verificare şi un protocol pentru **proba de autentificare**. Prezentarea sa în detaliu este:

Fie $p = 2q + 1$ un număr prim cu q prim, şi $\alpha \in Z_p^*$ un element de ordin q.
Pentru $1 \leq a_0 \leq q - 1$ se defineşte $\beta = \alpha^{a_0} \ (mod\ p)$.
Valorile p, q, α, β sunt publice şi considerate fixe.
Valoarea a_0 este generată de un arbitru şi este secretă pentru toată lumea (inclusiv pentru *Alice*).
Fie $\mathcal{P} = Z_q,\ \mathcal{A} = Z_q \times Z_q$.
O cheie este de forma

$$K = (\gamma_1, \gamma_2, a_1, a_2, b_1, b_2)$$

unde $a_1, a_2, b_1, b_2 \in Z_q,\ \gamma_1 = \alpha^{a_1}\beta^{a_2} \ (mod\ p), \quad \gamma_2 = \alpha^{b_1}\beta^{b_2} \ (mod\ p)$.
γ_1, γ_2 sunt publice, a_1, a_2, b_1, b_2 sunt secrete.
Dacă $x \in Z_q$, se defineşte $sig_K(x) = (y_1, y_2)$ unde

$$y_1 = a_1 + xb_1 \ (mod\ q), \quad y_2 = a_2 + xb_2 \ (mod\ q).$$

Pentru $y = (y_1, y_2) \in Z_q \times Z_q$, avem

$$ver_K(x, y) = T \quad \Longleftrightarrow \quad \gamma_1\gamma_2^x \equiv \alpha^{y_1}\beta^{y_2} \quad (mod\ p)$$

Se poate vedea direct că o semnătură construită corect este validată de funcţia de verificare.

Rămâne de studiat problema de securitate şi de ce procedeul este fără eşec.
Să stabilim întâi câteva proprietăţi importante ale cheilor.

Două chei $(\gamma_1, \gamma_2, a_1, a_2, b_1, b_2)$ şi $(\gamma_1', \gamma_2', a_1', a_2', b_1', b_2')$ sunt *echivalente* dacă

$$\gamma_1 = \gamma_1', \quad \gamma_2 = \gamma_2'.$$

În fiecare clasă de echivalenţă sunt exact q^2 chei.

Lema 3.1. *Dacă K, K' sunt chei echivalente, atunci*

$$ver_K(x, y) = T \Longleftrightarrow ver_{K'}(x, y) = T.$$

Demonstraţie. Fie $K = (\gamma_1, \gamma_2, a_1, a_2, b_1, b_2)$ şi $K' = (\gamma_1, \gamma_2, a_1', a_2', b_1', b_2')$ cu

$$\gamma_1 \equiv \alpha^{a_1}\beta^{a_2} \ (mod\ p) \equiv \alpha^{a_1'}\beta^{a_2'} \ (mod\ p), \qquad \gamma_2 \equiv \alpha^{b_1}\beta^{b_2} \ (mod\ p) \equiv \alpha^{b_1'}\beta^{b_2'} \ (mod\ p).$$

Să presupunem că mesajul x este semnat cu $y = (y_1, y_2)$ folosind cheia K, unde

$$y_1 \equiv a_1 + xb_1 \ (mod\ q), \qquad y_2 \equiv a_2 + xb_2 \ (mod\ q).$$

Dacă verificarea lui y se face cu cheia K', atunci:

$$\alpha^{y_1}\beta^{y_2} \equiv \alpha^{a_1' + xb_1'}\beta^{a_2' + xb_2'} \equiv \alpha^{a_1'}\beta^{a_2'}(\alpha^{b_1'}\beta^{b_2'})^x \equiv \gamma_1\gamma_2^x \quad (mod\ p).$$

Deci y se verifică şi cu cheia K. □

Lema 3.2. *Fie o cheie K şi $y = sig_K(x)$. Există exact q chei K' echivalente cu K astfel încât $y = sig_{K'}(x)$.*

Demonstraţie. Fie γ_1, γ_2 componentele publice ale lui K.
Trebuie determinat numărul de quadrupluri (a_1, a_2, b_1, b_2) astfel încât

$$\gamma_1 \equiv \alpha^{a_1}\beta^{a_2} \ (mod\ p), \qquad \gamma_2 \equiv \alpha^{b_1}\beta^{b_2} \ (mod\ p)$$

$$y_1 \equiv a_1 + xb_1 \ (mod\ q), \qquad y_2 = a_2 + xb_2 \ (mod\ q).$$

Cum α generează Z_q^*, există exponenţii unici $c_1, c_2, a_0 \in Z_q$ astfel încât

$$\gamma_1 \equiv \alpha^{c_1} \ (mod\ p), \quad \gamma_2 \equiv \alpha^{c_2} \ (mod\ p), \quad \beta \equiv \alpha^{a_0} \ (mod\ p).$$

În acest fel, este necesar şi suficient să avem:

$$c_1 \equiv a_1 + a_0a_2 \ (mod\ q), \qquad c_2 \equiv b_1 + a_0b_2 \ (mod\ q)$$

$$y_1 \equiv a_1 + xb_1 \ (mod\ q), \qquad y_2 \equiv a_2 + xb_2 \ (mod\ q).$$

Matricea acestui sistem de ecuaţii cu necunoscutele a_1, a_2, b_1, b_2 are rangul 3 (determinantul este $\begin{vmatrix} 1 & a_0 & 0 & 0 \\ 0 & 0 & 1 & a_0 \\ 1 & 0 & x & 0 \\ 0 & 1 & 0 & x \end{vmatrix}$), deci sistemul are cel puţin o soluţie netrivială obţinută cu ajutorul cheii K şi – în plus – dimensiunea spaţiului soluţiilor este $4 - 3 = 1$, deci există exact q soluţii. □

Lema 3.3. *Fie o cheie K, $y = sig_K(x)$ şi $ver_K(x', y') = T$ pentru $x' \neq x$. Există atunci cel puţin o cheie K' echivalentă cu K astfel ca*

$$y = sig_{K'}(x), \quad y' = sig_{K'}(x').$$

Demonstraţie. Se face printr-un raţionament analog cu cel din lema precedentă. □

Din aceste două leme putem trage următoarea concluzie: fiind dată o semnătură validă y a unui mesaj x, există exact q chei posibile care pot semna x.
Pentru orice alt mesaj $x' \neq x$, aceste q chei produc semnături diferite ale lui x'.

Se obţine astfel teorema următoare:

Teorema 3.4. *Fiind date $sig_K(x) = y$ şi $x' \neq x$, Oscar nu poate calcula $sig_K(x')$ decât cu probabilitate $1/q$.*

De remarcat că rezultatul acestei teoreme nu depinde de puterea de calcul a lui *Oscar*; securitatea provine din faptul că el nu poate distinge care din cele q chei posibile a fost utilizată.

Se poate explica acum noţiunea de semnătură fără eşec.
S-a arătat că fiind dat un mesaj x semnat cu y, *Oscar* nu poate calcula semnătura y' a lui *Alice* pe un alt mesaj x'.

Ar mai fi posibilitatea ca *Oscar* să poată calcula o semnătură $y" \neq sig_K(x')$ care să fie validă.
Dar, dacă ea ajunge înapoi la *Alice*, aceasta poate furniza cu probabilitate $1-1/q$ o *probă de autentificare*; aceasta este valoarea $a_0 = log_\alpha \beta$, cunoscută numai de arbitru.

Să presupunem că *Alice* este în posesia unei perechi (x', y'') astfel încât

$$ver_K(x', y'') = T \text{ şi } y'' \neq sig_K(x').$$

Avem

$$\gamma_1 \gamma_2^{x'} \equiv \alpha^{y_1{}''} \beta^{y_2{}''} \ (mod\ p)$$

unde $y'' = (y_1{}'', y_2{}'')$.

Alice poate calcula propria sa semnătură pentru x', pe care o notează $y' = (y_1{}', y_2{}')$ şi are

$$\gamma_1 \gamma_2^{x'} \equiv \alpha^{y_1'} \beta^{y_2'} \pmod p.$$

Deci $\alpha^{y''_1} \beta^{y''_2} \equiv \alpha^{y'_1} \beta^{y'_2} \ (mod\ p)$.
Scriind $\beta = \alpha^{a_0} \ (mod\ p)$ se obţine:

$$\alpha^{y''_1 + a_0 y''_2} \equiv \alpha^{y'_1 + a_0 y'_2} \ (mod\ p) \text{ de unde } y_1{}'' + a_0 y_2{}'' \equiv y'_1 + a_0 y'_2 \ (mod\ q)$$

sau $y_1{}'' - y'_1 \equiv a_0(y'_2 - y_2{}'') \ (mod\ q)$.
Evident $y'_2 \not\equiv y_2{}''$ deoarece y'' este un fals.

Deci $(y'_2 - y''_2)^{-1} \ (mod\ q)$ există şi avem:

$$a_0 = log_\alpha \beta = (y''_1 - y'_1)(y'_2 - y''_2)^{-1} \pmod q.$$

Bineînţeles, în verificarea probei de autentificare s-a presupus că nici *Alice* nu poate calcula logaritmul discret $log_\alpha \beta$.

Ca o remarcă finală, acest procedeu este cu utilizare unică, deoarece componentele secrete ale cheii K pot fi uşor determinate după două folosiri.

Exemplul 3.12. *Să presupunem* $p = 3467 = 2 \cdot 1733 + 1$. *Numărul* $\alpha = 4$ *are ordinul* 1733 *în* Z^*_{3467}.
Dacă se ia $a_0 = 1567$ *vom avea* $\beta = 4^{1567} \ (mod\ 3467) = 514$.

Reamintim că Alice cunoaşte α *şi* β *dar nu* a_0.

Să presupunem că Alice construieşte cheia sa folosind valorile

$$a_1 = 888, \quad a_2 = 1024, \quad b_1 = 786, \quad b_2 = 999$$

deci

$$\gamma_1 = 4^{888} \cdot 514^{1024} \ (mod\ 3467) = 3405 \quad \gamma_2 = 4^{786} \cdot 514^{999} \ (mod\ 3467) = 2281.$$

În acest moment Alice este pusă în prezenţa semnăturii false $(822, 55)$ *a mesajului* 3383.

Această semnătură este validă, deoarece condiţia de verificare este satisfăcută:

$$3405 \cdot 2281^{3383} \equiv 2282 \ (mod\ 3467) \quad 4^{822} \cdot 514^{56} \equiv 2282 \ (mod\ 3467).$$

Dar Alice ştie că aceasta nu este semnătura sa şi trebuie să dovedească acest lucru. Ea calculează propria sa semnătură:

$$(888 + 3383 \cdot 786 \ (mod\ 1733),\ 1024 + 3383 \cdot 999 \ (mod\ 1733)) = (1504, 1291)$$

după care evaluează logaritmul discret

$$a_0 = (822 - 1504)(1291 - 55)^{-1} \ (mod\ 1733) = 1567$$

care constituie probă de autentificare, şi arată că semnătura nu îi aparţine.

3.10 Protocoale de semnătură "blind"

Introdus de Chaum în 1983, protocolul de semnătură **blind** (*Blind Signature Protocol*) este extrem de util în construirea protocoalelor de comerţ electronic.

Vom presupune că *Alice* semnează un mesaj, iar *Bob* este entitatea care primeşte semnătura.

Definiţia 3.2. *Un protocol de semnătură* $(\mathcal{P}, \mathcal{A}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ *se numeşte "protocol de semnătură blind" dacă* $\forall x \in \mathcal{P}$ *(*x *complet necunoscut lui Alice) Bob poate să obţină de la Alice un mesaj semnat valid* $(x, sig_K(x))$.

În general un protocol de semnătură blind este format din:

- Un protocol de semnătură digitală $(\mathcal{P}, \mathcal{A}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ care aparţine lui *Alice*; $sig_A(x)$ reprezintă semnătura lui *Alice* asupra mesajului x.
- Două funcţii
$$f : \mathcal{P} \longrightarrow \mathcal{P}, \quad g : \mathcal{A} \longrightarrow \mathcal{A}$$
cunoscute doar de *Bob*, cu proprietatea
$$g(sig_A(f(m))) = sig_A(m)$$
f este numită "funcţie de orbire" (blinding function), iar g – funcţie de "dezvăluire" (unblinding function).

3.10.1 Protocolul de semnătură blind *RSA*

Fie p, q, n numere întregi pozitive definite ca la sistemul cu cheie publică RSA, şi

$$\mathcal{P} = \mathcal{A} = Z_n, \qquad \mathcal{K} = \{(n, p, q, a, b) \mid a, b \in Z_{\phi(n)},\ n = pq,\ \ ab \equiv 1\ (mod\ \varphi(n))\}$$

Fie m mesajul care trebuie semnat. Paşii protocolului blind sunt următorii:

1. *Bob* generează aleator un număr $r \in Z_n$ şi trimite lui *Alice* numărul
$$x' = r^b \cdot m\ (mod\ n)$$
2. *Alice* semnează mesajul m':
$y' = sig_K(m') = sig_K(r^b \cdot m) = (r^b \cdot m)^a\ (mod\ n) \equiv r \cdot m^a \quad (mod\ n)$
şi trimite y' lui *Bob*.
3. *Bob* calculează
$$y = y' \cdot r^{-1} = x^a\ (mod\ n).$$

La sfârşitul protocolului, *Bob* obţine $(x, y = sig_K(m))$ – un mesaj semnat valid, astfel încât *Alice* nu cunoaşte m.

Observăm ca funcţiile sig_K şi ver_K sunt cele definite în protocolul de semnătură RSA (Exemplul 3.1):

$sig_K(m') = (m')^a \quad (mod\ n)$
$ver_K(m', y') = T \quad \Longleftrightarrow \quad m' = (y')^b \quad (mod\ n).$

Numărul r se numeşte *factor de orbire (blinding factor)*; rolul lui este de a ascunde mesajul m.

Exemplul 3.13. *Să presupunem că* $n = 77$ *şi Alice are cheile* $a = 7$, $b = 43$.
Bob vrea să obţină o semnătură blind a lui Alice pe mesajul $m = 14$ *şi alege valoarea aleatoare* $r = 5$. *El va calcula*

$$x' = 14^7 \cdot 5 \equiv 31 \quad (mod\ 77)$$

Alice primeşte valoarea $x' = 31$ *şi semnează:*

$$y' = (x')^a = 31^{43} = 3 \quad (mod\ 77)$$

Din această valoare, Bob deduce semnătura $3 \cdot 5^{-1} = 5 \cdot 31 = 16\ \ (mod\ 77)$.
Perechea $(14, 16)$ *este un mesaj semnat valid.*

Vom prezenta alte protocoale de semnătură blind în Capitolul 7, destinat sistemelor electronice de plată.

3.11 Semnături cu mandat

Noţiunea de semnătură cu mandat (*Proxy Signature*) este introdusă în 1996 de Mambo şi Okamoto ([52]). Ea permite convertirea unei semnături $sig_{Alice}(x)$ pentru un mesaj $x \in \mathcal{P}$ într-o semnătură validă $sig_{Bob}(x)$, fără a dezvălui informaţii legate de cele două protocoale de semnătură. Formal

Definiţia 3.3. *Fiind date două entităţi A_1, A_2 având protocoalele de semnătură $(\mathcal{P}, \mathcal{A}_1, \mathcal{K}, \mathcal{S}_1, \mathcal{V}_1)$ respectiv $(\mathcal{P}, \mathcal{A}_2, \mathcal{K}, \mathcal{S}_2, \mathcal{V}_2)$, un protocol de semnătură cu mandat este o funcţie $sig_{\pi_{A_1 \to A_2}} : \mathcal{A}_1 \times \mathcal{P} \longrightarrow \mathcal{A}_2 \times \mathcal{P}$ definită*

$$sig_{\pi_{A_1 \to A_2}}(sig_{A_1}(x)) = sig_{A_2}(x).$$

Cheia $\pi_{A_1 \to A_2}$ se numeşte "cheie mandatată".

Pe lângă condiţiile obişnuite pe care trebuie să le verifice un protocol de semnătură, o semnătură cu mandat trebuie să fie **inatacabilă (unforgeability)**: deşi *Oscar* are acces la predicatele ver_{Alice}, ver_{Bob}, la cheile mandatate, la mesaje semnate (chiar prin atacuri cu text clar ales), el nu poate crea semnături pentru mesaje noi (deci nu poate obţine informaţii despre funcţiile sig_{Alice} şi sig_{Bob}).

De remarcat că o semnătură cu mandat este o semnătură blind: *Bob* primeşte mandat să semneze un mesaj x, dar el *nu are acces* la acest mesaj.

Clasificări ale semnăturilor cu mandat:

- *Încredere între parteneri*: Evident, dacă *Bob* primeşte mandat să transforme o semnătură a lui *Alice* în propria sa semnătură, el are încredere în *Alice* (deoarece acceptă să semneze mesaje pe care nu le cunoaşte). Dar nu este necesar ca *Alice* să aibă încredere în *Bob*. Mai exact, o semnătură cu mandat este *simetrică* dacă din cheia mandatată $\pi_{Alice \to Bob}$, *Bob* poate obţine funcţia de semnătură sig_{Alice}. În caz contrar, semnătura cu mandat este *asimetrică*.

- *Acces la calculul unei chei mandatate*: Un protocol de semnătură asimetrică cu mandat este *activ* dacă obţinerea cheii mandatate este posibilă numai dacă *Alice* foloseşte funcţia sig_{Alice}. Dacă pentru calculul acestei chei nu este necesară funcţia de semnătură a lui *Alice* (ci numai predicatul ver_{Alice}), atunci protocolul este *pasiv*.

- *Anonimitatea cheilor mandatate*: Semnăturile cu mandat mai pot fi clasificate după cantitatea de informaţie pe care o dezvăluie relativ la predicatele de verificare ale celor două entităţi implicate în procesul de semnătură.
  Astfel, un protocol de semnătură cu mandat este

  – *transparent*, dacă cheia mandată $\pi_{Alice \to Bob}$ dezvăluie detaliile proceselor de verificare ale ambilor participanţi.

  – *translucidă*, dacă un intrus care ghiceşte detaliile proceselor de verificare poate verifica aceste informaţii utilizând cheia mandatată $\pi_{Alice \to Bob}$.

– *opacă* dacă o cheie mandatată nu oferă nici o informaţie despre procesele de verificare ale semnăturilor lui *Alice* şi *Bob*.

3.11.1 Semnătura cu mandat *El Gamal*

Este o variantă a protocolului de semnătură *El Gamal*.

Fie p un număr prim mare, q un divizor prim mare al lui $p-1$ şi $\alpha \in Z_p^*$ un element de ordin q.

Alice are pentru semnătură perechea (x_A, y_A) unde $x_A \in Z_q^*$ este secretă, iar $y_A = \alpha^{x_A} \pmod p$ este publică.
Similar, *Bob* dispune de perechea (x_B, y_B).

$h : Z_2^* \longrightarrow Z_q$ este o funcţie de dispersie criptografică.

Alice semnează un mesaj $m \in Z_p^*$ alegând aleator o valoare $k \in Z_q^*$ şi calculând semnătura $sig_A(m) = (\gamma, \delta)$ unde

$$\gamma = (\alpha^k \pmod p) \pmod q, \qquad \delta = x_A\gamma + k \cdot h(m)) \pmod q$$

Procedura de verificare este definită

$$ver_A(m, (\gamma, \delta)) = T \quad \Longleftrightarrow \quad \alpha^\delta = y_A^\gamma \cdot \gamma^{h(m)} \pmod p$$

Pentru a stabili un mandat de semnătură între ei, *Alice* şi *Bob* calculează cheia mandatată

$$\pi_{A\to B} = x_B - x_A$$

Dacă *Alice* şi *Bob* nu au încredere unul în altul, ei pot obţine această cheie printr-un protocol.
Un exemplu de astfel de protocol poate fi (toate calculele se efectuează modulo p):

1. *Alice* generează aleator $u \in Z_p$ şi trimite lui *Bob* $K = x_A + u$;
2. *Bob* generează aleator numărul $v \in Z_p$ şi trimite lui *Alice* valoarea $K \longleftarrow K - x_B + v$;
3. *Alice* calculează $K \longleftarrow K - u$ şi trimite lui *Bob* noua valoare;
4. *Bob* determină cheia mandatată prin $\pi_{A\to B} = v - K$.

Dacă $sig_A(m) = (\alpha^k, x_A \cdot \gamma + k \cdot h(m))$ este semnătura lui *Alice* pe mesajul m, atunci *Bob* poate să o transforme în semnătura sa astfel:

$$sig_B(m) = (\gamma, \delta + \gamma \cdot \pi_{A\to B}) = (\alpha^k, x_A \cdot g + k \cdot h(m) + (x_B - x_A) \cdot \gamma) = (\alpha^k, x_B \cdot \gamma + k \cdot h(m))$$

De remarcat că schema este simetrică, pentru că *Bob* poate deduce x_A din cheia mandatată $\pi_{A\to B}$ şi din propria sa cheie secretă x_B.
De asemenea, schema este activă deoarece calculul cheii mandatate necesită cheia secretă x_A a lui *Alice*.

3.11.2 Semnătură cu mandat bazată pe curbe eliptice

Protocolul de semnătură cu mandat prezentat în această secţiune a fost propus în 2002 de Z. Tan, Z. Liu şi C. Tang ([79]). Aici *Alice* foloseşte o semnătură blind a lui *Bob* – generată pe baza unei chei mandatate – pentru a crea semnătura unui mesaj.

Fie E o curbă eliptică şi P un generator al grupului aditiv al punctelor de pe E. Presupunem că ordinul lui P este n. Vom nota cu Q_x abscisa unui punct Q de pe curba E, iar h este o funcţie de dispersie criptografică.

Fie $k_A,\ k_B \in Z_n$ cheile secrete ale lui *Alice* respectiv *Bob* şi $P_A = k_A P,\ P_B = k_B P$ cheile publice.

Deci E, P, n, P_A, P_B, h sunt publice.

A. *Faza de delegare*:

1. *Alice* generează aleator $k \in Z_n^*$ şi calculează
$$R = kP, \quad r = R_x, \quad s = k_A r + k \ (mod\ n)$$
2. Trimite lui *Bob* tripletul (r, s, R);
3. *Bob* verifică egalitatea $R = sP - rP_A$.
4. Dacă relaţia este satisfăcută, *Bob* calculează $s' = s + k_B \ (mod\ n)$.

B. *Faza de semnătură cu mandat*:

1. *Bob* generează aleator $u \in Z_n^*$ şi calculează $T = uP$;
2. Trimite lui *Alice* valoarea T;
3. *Alice* generează aleator $a, b \in Z_n$ şi calculează
$$\begin{aligned} R' &= T + bP + (-a-b)P_B + (-a)R + (-ar)P_A, \\ \alpha &= h(R'_x \| m) \pmod n, \\ \alpha^* &= \alpha - a - b \pmod n, \\ U &= (-\alpha + b)R + (-\alpha + b)rP_A - \alpha P_A \end{aligned}$$
4. Trimite lui *Bob* valoarea α^*.

5 *Bob* calculează $x = \alpha^* \cdot s' + u \pmod n$ şi trimite această valoare lui *Alice*;

6 *Alice* calculează $\beta = x + b \pmod n$

Semnătura este $sig(m) = (\alpha, \beta, U)$. Protocolul de verificare este

$$ver(m, (\alpha, \beta, U)) = T \iff \alpha = h((\beta P - \alpha P_B + \alpha P_A + U)_x \| m) \pmod n$$

3.12 Semnături de grup

Conceptul de semnătură de grup este definit de D. Chaum şi E. Heyst ([15]) în 1991. El permite membrilor unui grup să semneze mesaje în numele grupului, astfel ca:

- Numai membrii grupului pot genera semnături valide ale mesajelor.
- Semnăturile pot fi verificate cu o singură cheie publică.
- Este imposibil să se afle ce membru al grupului a semnat mesaje sau dacă două semnături au fost generate de acelaşi membru al grupului.
- Există un manager al grupului care poate – în caz de dispută – să afle identitatea semnatarului.
- Membrii grupului nu pot bloca o acţiune de dezvăluire din partea managerului.
- Nimeni (nici chiar managerul) nu poate semna în locul altui membru al grupului.

Semnăturile de grup pot fi utilizate de exemplu de o companie pentru a autentifica o listă de preţuri, pentru comunicate de presă sau pentru contracte digitale. Partenerii au acces la o singură cheie publică pentru verificarea semnăturii, iar compania poate să ascundă structura internă şi respnsabilităţile angajaţilor săi, putând totuşi – în caz de litigiu – să determine ce angajat a semnat un anumit document.

O semnătură de grup este compusă din cinci faze:

1. **Iniţializare**: Un algoritm prin care managerul stabileşte cheia publică Y a grupului şi cheia sa secretă de administrare.
2. **Înscriere**: Un protocol interactiv stabilit între managerul grupului şi fiecare membru P al grupului. Scopul ei este stabilirea cheii secrete x a lui P.
3. **Semnare**: Un algoritm care, pentru un mesaj m şi o cheie secretă x, determină o semnătură validă $s = sig_x(m)$.

4. **Verificare**: Un algoritm care, pentru un mesaj m, o semnătură s şi cheia publică Y a grupului, decide validitatea semnăturii.

5. **Deschidere**: Un algoritm care, pentru o semnătură s şi cheia de administrare a managerului, determină identitatea membrului din grup care a produs semnătura s.

Se presupune că toate canalele de comunicaţie între membrii grupului şi manager sunt sigure.

De remarcat că managerul nu cunoaşte cheile secretele ale membrilor grupului (altfel, ar putea semna mesaje în locul acestora).

Pot fi găsite în literatură numeroase de scheme de semnături de grup. Toate încearcă să îmbunătăţească o serie de parametri (lungimea cheii publice de grup, lungimile semnăturilor, eficienţa algoritmilor şi protocoalelor) sau să rezolve diverse situaţii particulare (semnături blind de grup, semnături bazate pe ID, semnături fără manager etc). Vom prezenta două astfel de scheme de semnătură, ambele folosind curbe eliptice.

3.12.1 Semnătură de grup bazată pe ID

Considerăm o curbă eliptică E şi:

- Un punct P pe curbă, de ordin q (q număr prim mare); fie $G_1 =< P >$ grupul aditiv generat de P.
- Un grup multiplicativ G_2 de ordin q.
- O pereche biliniară $e : G_1 \times G_1 \longrightarrow G_2$.
- Două funcţii de dispersie criptografică

$$h_1 : \{0,1\}^* \times G_1 \longrightarrow Z_q, \qquad h_2 : \{0,1\}^* \times G_1 \longrightarrow G_1$$

Securitatea protocolului se bazează pe următoarele probleme (a se vedea şi Anexa 2):

1. $ECDLP$ (Problema Logaritmului Discret pe Curbe eliptice): Fiind date elementele P (generator al grupului G_1) şi $Q \in G_1$, să se găsească un număr $n \in Z_q^*$ astfel ca $Q = nP$. Pentru q suficient de mare, aceasta este o problemă $\mathcal{NP}$ - completă.

2. $CDHP$ (Problema Diffie - Hellman computaţională): Fiind date generatorul P şi $aP,\ bP \in G_1$ unde $a, b \in Z_q^*$ sunt arbitrare, să se determine elementul abP.
Şi aceasta este o problemă $\mathcal{NP}$ - completă.

3. $DDHP$ (Problema Diffie - Hellman decizională): Fiind date generatorul P şi aP, bP, cP unde $a, b, c \in Z_q$, să se decidă dacă $c \equiv ab \ (mod\ q)$.

Se consideră – în cazul componentelor definite mai sus – că problema $DDHP$ este rezolvabilă în timp polinomial, iar pentru celelalte două probleme nu există algoritmi de rezolvare în timp polinomial.

Protocoalele schemei de semnătură sunt:

1. **Iniţializare:**

   Managerul grupului generează aleator $s \in Z_q^*$ şi calculează $Q = sP$.
   Parametrii publici sunt $\{G_1, G_2, e, q, P, Q, h_1, h_2\}$, iar s este cheia secretă.

2. **Înscriere**: *Alice* doreşte să devină membru al grupului. Pentru aceasta:

   (a) *Alice* generează un număr aleator $r \in Z_q^*$, calculează $P' = rP$ şi trimite mana-gerului perechea (P', ID) unde ID este o secvenţă de informaţii (codificate în binar) relative la identitatea sa.

   (b) Managerul calculează $Q_{ID} = h_2(ID\|T, P')$ şi $S_{ID} = s \cdot Q_{ID}$, unde T este durata de valabilitate a valorii r. Valoarea S_{ID} este trimisă lui *Alice*.
   Cheia privată de utilizator a lui *Alice* este (S_{ID}, r), iar cheia publică este ID.

   (c) *Alice* generează aleator $x \in Z_q^*$ şi trimite managerului (xP', xP, S_{ID}).
   Managerul verifică egalităţile

   $$S_{ID} = s \cdot h_2(ID\|T, P'), \qquad e(xP', P) = e(xP, P').$$

   Dacă aceste egalităţi sunt verificate, managerul trimite lui *Alice* valoarea

   $$S = s \cdot h_2(T, xP')$$

   şi adaugă la lista de membri secvenţa (ID, xP, P', xP') corespunzătoare noului membru.

   (d) *Alice* are certificatul de membru de grup (S, xP') şi cheia privată de semnătură $r \cdot x$. De remarcat că valorile r şi x nu le ştie decât *Alice*.

3. **Semnare**: Pentru a semna mesajul m, *Alice* urmează algoritmul:

   (a) Generează aleator $a \in Z_q^*$.

   (b) Calculează

   $$U = a \cdot h_2(T, xP'), \qquad V = r \cdot x \cdot h_2(m, U),$$
   $$H = h_1(m, U + V), \qquad W = (a + H) \cdot S.$$

   (c) Semnătura mesajului m este (U, V, W, T, xP').

4. **Verificare**: Dacă perioada T este validă, semnătură este verificată astfel:

   (a) Se calculează $\alpha = h_2(T, xP'), \quad H = h_1(m, U + V)$.

   (b) Se verifică
   $$e(W, P) = e(U + H\alpha,\ Q),$$
   $$e(V, P) = e(h_2(m, U),\ xP')$$

5. **Deschidere**: Managerul extrage din semnătură componenta $xP' = rxP$.
   Cu ajutorul ei poate identifica pe *Alice* din lista membrilor grupului.
   Alice nu-şi poate repudia semnătura, deoarece managerul poate demonstra că semnătura îi aparţine:
   $$e(xP', P) = e(xP, P'),$$
   $$e(S_{ID}, P) = e(h_2(ID\|T, P'), Q)$$

Consistenţa relaţiilor de la $(4b)$:
$e(U+H\alpha,\ Q) = e(a\alpha+H\alpha, sP) = e((a+H)\alpha, sP) = e((a+H)s\alpha, P) = e((a+H)S, P) = e(W, P)$,
$e(h_2(m, U),\ xP') = e(h_2(m, U), rxP) = e(rx \cdot h_2(m, U), P) = e(V, P)$.

Cu cele două relaţii de non-repudiere de la etapa de deschidere, managerul arată că informaţiile (ID, P', xP, xP') cu care este înscrisă *Alice* sunt consistente cu componentele publice P, Q, cu cheia secretă S_{ID} a lui *Alice* şi cu informaţia secretă Q_{ID} deţinută de manager.

De remarcat că managerul nu poate semna în locul lui *Alice* decât dacă poate calcula elementele $aP, bP, cP \in G_1$ cu $a \equiv bc\ (mod\ q)$. Chen, Zhang şi Kim ([19]) definesc această problemă $RCDHP$ (Reversion of $CDHP$) şi arată că

Teorema 3.5. *În G_1, problemele $RCDHP$ şi $CDHP$ sunt echivalente.*

Demonstraţie.

i. $RCDHP \Longrightarrow CDHP$: Din calculul lui bP, cP, P rezultă $1 \equiv bc\ (mod\ q)$ sau $c = b^{-1}$. Deci s-a calculat $b^{-1}P$.
Din $aP, b^{-1}P, cP$ rezultă $a \equiv c \cdot b^{-1}\ (mod\ q)$. Cum $a \equiv (ab)b^{-1}\ (mod\ q)$, avem $c = ab$.
Deci, din aP, bP am obţinut abP, adică s-a rezolvat $CDHP$.

ii. $CDHP \Longrightarrow RCDHP$: Fie $Q = bP$, sau – altfel scris – $p = b^{-1}Q$.
Din $b^{-1}Q,\ b^{-1}Q$ rezultă $b^{-2}Q = b^{-1}P$.
Din $aP, b^{-1}P$ rezultă $ab^{-1}P$.
Deci s-au putut calcula $aP, b^{-1}P, ab^{-1}P$ – soluţia problemei $RCDHP$. □

3.12.2 Semnătură multiplă de grup, fără manager

În unele situaţii, existenţa unui manager de grup este inutilă, imposibilă sau chiar periculoasă pentru securitatea sistemului. De aceea, uneori sunt necesare astfel de semnături de grup, construite fără prezenţa unui manager. Schema prezentată aici a fost propusă de T. Chen, K. Huang şi Y. Chung ([18]) şi se referă la semnarea unui mesaj de către toţi membrii grupului.

1. **Iniţializare**: Toţi cei N membri ai grupului cad de acord asupra următorilor parametri:
   (a) Un număr mare q – prim sau putere a lui 2;
   (b) Parametrii $a, b \in Z_q$, $4a^3 + 27q^2 \neq 0 \ (mod\ q)$, care definesc curba eliptică
   $$E: \ y^2 = x^3 + ax + b \quad (mod\ q)$$
   (c) Un punct $B = (x_b, y_b) \in E(Z_q)$ de ordin n, unde n este număr prim mare.
   (d) $B_1, B_2 \in E(Z_q)$, două puncte de ordin n.
   (e) Un parametru de securitate t (se consideră $t \geq 72$).
2. **Generare chei**: Fiecare membru U_i $(1 \leq i \leq N)$:

   (a) Generează aleator $d_{i_1}, d_{i_2} \in Z_n^*$; perechea (d_{i_1}, d_{i_2}) formează cheia sa privată.
   (b) Calculează cheia publică
   $$Y_i = d_{i_1} B_1 + d_{i_2} B_2$$
   pe care o trimite celorlaltor membri ai grupului.
   (c) Determină cheia publică de grup
   $$Y = \sum_{i=1}^{N} Y_i.$$

3. **Semnare**: Pentru semnarea mesajului m, fiecare membru U_i al grupului urmează algoritmul:

   (a) Generează aleator $r_{i_1}, r_{i_2} \in [1, 2^n]$, cu ajutorul cărora construieşte punctul
   $$Q_i = r_{i_1} B_1 + r_{i_2} B_2.$$
   (b) Trimite Q_i celorlaltor membri ai grupului, după care determină
   $$Q = \sum_{i=1}^{N} Q_i.$$
   (c) Folosind o funcţie de dispersie h, calculează
   $$e = h(m, Q) \in [1, 2^t]$$

(d) Calculează valorile s_{i_1} şi s_{i_2} după formulele
$$s_{i_1} = (r_{i_1} + e \cdot d_{i_1}) \ (mod\ n), \qquad s_{i_2} = (r_{i_2} + e \cdot d_{i_2}) \ (mod\ n)$$

(e) Trimite perechea (s_{i_1}, s_{i_2}) celorlaltor membri ai grupului, după care calculează
$$s_1 = \sum_{i=1}^{N} s_{i_1} \ (mod\ n), \qquad s_2 = \sum_{i=1}^{N} s_{i_2} \ (mod\ n)$$

(f) Semnătura mesajului m este (e, s_1, s_2).

4. **Verificare**: Dacă *Bob* doreşte să verifice validitatea semnăturii (e, s_1, s_2) a mesajului m, va proceda astfel:

   (a) Calculează $Z = s_1B_1 + s_2B_2 - eY$.

   (b) Dacă $e = h(m, Z)$, atunci acceptă semnătura drept validă.

Teorema 3.6. *Dacă schema de mai sus este urmată corect de cei N membri ai grupului, atunci semnătura obţinută este validă.*

Demonstraţie. Se efectuează calculele

$$Z = s_1B_1 + s_2B_2 - eY = \left(\sum_{i=1}^{N} s_{i_1} \ (mod\ n)\right) B_1 + \left(\sum_{i=1}^{N} s_{i_2} \ (mod\ n)\right) B_2 - eY =$$
$$= \left[\left(\sum_{i=1}^{N} r_{i_1}\right) B_1 + \left(e\sum_{i=1}^{N} d_{i_1}\right) B_1\right] + \left[\left(\sum_{i=1}^{N} r_{i_2}\right) B_2 + \left(e\sum_{i=1}^{N} d_{i_2}\right) B_2\right] - eY =$$
$$= \left[\left(\sum_{i=1}^{N} r_{i_1}\right) B_1 + \left(\sum_{i=1}^{N} r_{i_2}\right) B_2\right] + \left[\left(e\sum_{i=1}^{N} d_{i_1}\right) B_1 + \left(e\sum_{i=1}^{N} d_{i_2}\right) B_2\right] - eY =$$
$$= \sum_{i=1}^{N}(r_{i_1}B_1 + r_{i_2}B_2) + e\sum_{i=1}^{N}(d_{i_1}B_1 + d_{i_2}B_2) - eY = \sum_{i=1}^{N} Q_i + e\sum_{i=1}^{N} Y_i - eY =$$
$$= Q + eY - eY = Q. \qquad \square$$

Relativ la securitatea acestei scheme,autorii au luat în considerare trei atacuri posibile:

- *Oscar* vrea să ia locul lui U_i, deci să determine o cheie privată (d_{i_1}, d_{i_2}) folosind Y_i sau (s_{i_1}, s_{i_2}).
- *Oscar* caută să genereze o semnătură validă (e, s_1, s_2) pentru un mesaj m.
- Un membru al grupului vrea să genereze o semnătură validă pe un mesaj m, fără a coopera cu ceilalţi membri.

Toate aceste atacuri se reduc la rezolvarea $ECDLP$, o problemă $\mathcal{NP}$ - completă.

3.13 Datare

Problema *datării* a fost introdusă şi studiată în [3] şi [39]. Ea se referă la faptul că, datorită timpului lung cât poate fi utilizat un document, orice semnătură poate fi spartă. Odată ce *Oscar* descoperă algoritmul de semnătură, este pusă în discuţie autenticitatea tuturor mesajelor.

Un alt scenariu nedorit este următorul: *Alice* semnează un mesaj, apoi publică secretul semnăturii şi o reneagă declarând că a fost construită de altcineva.

Toate aceste probleme pot fi rezolvate dacă se introduce şi un procedeu de datare a semnăturii, care constituie o probă că un document a fost semnat la o anumită dată. În acest fel, dacă secretul lui *Bob* a fost compromis, aceasta nu afectează semnăturile anterioare. Principiul cărţilor de credit este similar: o carte de credit anunţată unei bănci ca pierdută, este anulată fără ca aceasta să afecteze plăţile anterioare făcute pe baza ei.

Vom prezenta aici câteva procedee de datare.

În primul rând, *Alice* poate construi singură o datare. Pentru aceasta, ea poate începe prin a colecta un anumit număr de informaţii publicate recent (care nu ar fi putut fi prezise). De exemplu, poate da rezultatele unei etape de fotbal, rata de schimb euro/leu la o anumită bancă, etc.
Vom nota această informaţie *inf.pub*.

Să presupunem acum că *Alice* vrea să dateze semnătura unui mesaj x. Dacă h este o funcţie de dispersie criptografică (publică), *Alice* va urma algoritmul următor:

1. Calculează $z = h(x)$;
2. Calculează $z' = h(z \| inf.pub)$;
3. Determină $y = sig_K(z')$;
4. Publică într-un ziar din ziua următoare $(z, inf.pub, y)$.

Prezenţa informaţiei *inf.pub* arată că *Alice* nu a putut calcula y înainte de data în discuţie. Faptul că y este publicat într-un cotidian de a doua zi asigură faptul că *Alice* nu a putut calcula y după data apariţiei ziarului. Astfel, data semnăturii lui *Alice* este restrânsă la un interval de o zi.
De remarcat că *Alice* nu dezvăluie mesajul x, deoarece a fost publicat numai z.

Dacă există un serviciu oficial de datare (un fel de notar public electronic), se poate construi un alt procedeu:

Alice calculează $z = h(x)$ şi $y = sig_K(z)$, apoi supune (z, y) serviciului de datare.

Acesta adaugă o dată D şi semnează tripletul (z, y, D).

În acest fel, *Alice* poate certifica că a semnat înainte de o anumită dată. Pentru a arăta că a semnat după o anumită dată, ea poate adăuga în semnătură o anumită informaţie

publică *inf.pub*.

Este posibil ca *Alice* să nu aibă încredere în serviciul public de datare. Atunci ea poate creşte securitatea legând secvenţial toate mesajele datate. Mai detaliat:

Alice trimite serviciului de datare un triplet (z, y, ID_{Alice}), unde z este amprenta numerică a lui x, y este semnătura sa, iar ID_{Alice} este o informaţie care o identifică pe *Alice*.

Serviciul va data secvenţa de astfel de triplete.
Dacă (z_n, y_n, ID_n) este al n-lea triplet, fie t_n data atribuită lui.

Serviciul va data acest triplet după algoritmul:

1. Calculează $L_n = (t_{n-1}, ID_{n-1}, z_{n-1}, y_{n-1}, h(L_{n-1}))$;
2. Calculează $C_n = (n, t_n, z_n, y_n, ID_n, L_n)$;
3. Determină $s_n = sig_{serviciu}(h(C_n))$;
4. Înapoiază tripletul (C_n, s_n, ID_{n+1}) lui ID_n.

Aici L_n reprezintă informaţia *de legătură* între tripletele n şi $n-1$ (L_0 este fixat arbitrar la iniţializarea protocolului).

3.14 Exerciţii

3.1. *Să presupunem că Alice utilizează semnătura El Gamal şi semnează mesajele x_1, x_2 obţinând (γ, δ_1) respectiv (γ, δ_2) (cu aceeaşi valoare a lui γ în ambele semnături).*
Se consideră în plus că $(\delta_1 - \delta_2, p-1) = 1$.

- *Arătaţi că aceste informaţii sunt suficiente pentru determinarea lui k;*
- *Arătaţi cum se poate sparge protocolul de semnătură;*
- *Presupunând $p = 3187$, $\alpha = 5$, $\beta = 25703$, efectuaţi calculul lui k şi a plecând de la semnăturile $(23972, 31396)$ pentru $x = 8990$ şi $(23972, 20481)$ pentru $x = 31415$.*

3.2. *Protocolul de semnătură El Gamal este implementat folosind $p = 31847$, $\alpha = 5$, $\beta = 26379$. Să se scrie un program care:*

- *Verifică semnătura $(20679, 11082)$ a mesajului $x = 20543$.*
- *Calculează exponentul secret a prin compromisul spaţiu - timp al lui Shanks ([2], pag. 165). Apoi determină valoarea aleatoare k utilizată în semnătura lui x.*

3.3. *Alice utilizează protocolul de semnătură El Gamal ca în Exemplul 3.4:* $p = 467$, $\alpha = 2$, $\beta = 132$. *Să presupunem că ea semnează mesajul* $x = 100$ *cu* $(29, 51)$. *Calculaţi semnătura falsă pe care o poate obţine Oscar cu* $h = 102$, $i = 45$, $j = 293$. *Autentificaţi semnătura rezultată cu funcţia de verificare.*

3.4. *Arătaţi că a doua metodă de atac din semnătura El Gamal furnizează o semnătură corectă care satisface funcţia de verificare.*

3.5. *Modificăm puţin protocolul de semnătură El Gamal. Cheia este construită astfel: Alice alege o rădăcină primitivă* $\alpha \in Z_p^*$, *un exponent secret* $a \in Z_{p-1}$, $(a, p-1) = 1$ *şi* $\beta = \alpha^a$. *Cheia este* $K = (\alpha, a, \beta)$ *unde* α, β *sunt publice, iar* a *este secretă. Fie* $x \in Z_p$ *un mesaj care trebuie semnat. Alice calculează semnătura* $sig_K(x) = (\gamma, \delta)$ *prin:*

$$\gamma = \alpha^k \ (mod\ p) \quad \delta = (x - k\gamma)a^{-1} \ (mod\ (p-1)).$$

Singura diferenţă faţă de semnătura El Gamal este calculul lui δ.

- *Descrieţi cum se poate verifica cu cheia publică a lui Alice o semnătură* (γ, δ) *pe un mesaj* x;
- *Descrieţi avantajul noului procedeu (faţă de cel vechi) din punct de vedere al complexităţii calculelor;*
- *Comparaţi pe scurt securitatea celor două protocoale.*

3.6. *Alice utilizează protocolul DSA cu* $q = 101$, $p = 7879$, $\alpha = 170$, $a = 75$, $\beta = 4567$. *Determinaţi semnătura lui Alice pe mesajul* $x = 5001$ *utilizând valoarea* $k = 49$, *şi arătaţi cum poate fi verificată semnătura rezultată.*

3.7. *În protocolul de semnătură Lamport, Alice semnează două mesaje* x, x', *ambele de câte* k *biţi. Fie* $s = d(x, x')$ *numărul de coordonate în care diferă cele două mesaje. Arătaţi că Oscar poate semna* $2^s - 2$ *mesaje noi.*

3.8. *În protocolul de semnătură Bos-Chaum cu* $k = 6$, $n = 4$ *sunt semnate mesajele* $x = (0, 1, 0, 0, 1, 1)$, $x' = (1, 1, 0, 1, 1, 1)$. *Determinaţi noile mesaje pe care le poate semna Oscar, plecând de la semnăturile lui* x *şi* x'.

3.9. *În protocolul de semnătură Bos-Chaum, Alice semnează două mesaje* x, x'. *Fie* $s = card(\phi(x) \cup \phi(x'))$. *Arătaţi că Oscar poate semna acum* $C_s^n - 2$ *mesaje noi.*

3.10. *Alice utilizează protocolul de semnătură incontestabilă Chaum-van Antwerpen ca în Exemplul* 3.10*; deci* $p = 467$, $\alpha = 4$, $a = 101$, $\beta = 449$. *Să presupunem că Alice este confruntată cu semnătura* $y = 25$ *a mesajului* $x = 157$ *şi doreşte să arate că este falsă. Presupunând că Bob alege valorile aleatoare* $e_1 = 46$, $e_2 = 123$, $f_1 = 198$, $f_2 = 11$ *în protocolul de dezminţire, calculaţi întrebările* c, d *ale lui Bob şi răspunsurile* C, D *ale lui Alice; verificaţi că Bob admite dezminţirea.*

3.11. *Alice utilizează protocolul de semnătură fără eşec Pedersen - van Heyst cu* $p = 3467,\ \alpha = 4,\ a_0 = 1567,\ \beta = 514$ *(valoarea lui* a_0 *nu este cunoscută de Alice).*

- *Folosind faptul că* $a_0 = 1567$*, determinaţi toate cheile* $K = (\gamma_1, \gamma_2, a_1, a_2, b_1, b_2)$ *astfel ca* $sig_K(42) = (1118, 1449)$*.*
- *Presupunem* $sig_K(42) = (1118, 1449)$ *şi* $sig_K(969) = (899, 471)$*. Fără a utiliza valoarea lui* a_0*, determinaţi valoarea lui* K *(din protocolul cu cheie One - Time).*

3.12. *Alice foloseşte protocolul de semnătură fără eşec Pedersen - van Heyst cu* $p = 5087,\ \alpha = 25,\ \beta = 1866$*. Cheia este* $K = (5065, 5076, 144, 874, 1873, 2345)$*. Se presupune că Alice este confruntată cu semnătura* $(2219, 458)$ *contrafăcută pe mesajul* 4785*.*

- *Arătaţi că semnătura satisface condiţia de verificare (deci este validă).*
- *Arătaţi cum poate Alice să calculeze proba de autenticitate plecând de la această semnătură.*

Capitolul 4

Sisteme de partajare a secretelor

Vom începe cu două exemple simple:

- Într-o bancă, seiful trebuie deschis în fiecare zi. Banca are trei directori, dar nu încredinţează combinaţia seifului nici unuia din ei. Ea doreşte să dispună de un sistem de acces prin care orice asociere de doi directori să poată deschide seiful, dar acest lucru să fie imposibil pentru unul singur.

- Conform revistei *Time Magazin* (4 mai 1992), în Rusia, accesul la arma nucleară utilizează un sistem *doi - din - trei* similar. Cele trei persoane sunt Preşedintele ţării, Preşedintele Parlamentului şi Ministrul Apărării.

Vom face o primă formalizare:
Fiind dat un secret S, se cere împărţirea lui la n participanţi ($n \geq 2$), astfel încât:

1. Cel puţin k din cei n participanţi pot regăsi S prin combinarea informaţiile lor;

2. Nici o asociere de mai puţin de k participanţi nu pot recompune S.

Acest lucru se poate realiza prin *partajarea* secretului S în n componente ("shares" în engleză) $S_1, S_2, \ldots, S_n$ şi distribuirea câte unei componente fiecărui participant.

Intuitiv, un sistem de partajare a secretelor este deci o metodă de spargere a unui secret în componente, astfel încât acesta să poată fi recompus numai de către *grupurile autorizate* (grupuri care au dreptul să refacă secretul).

În funcţie de "cantitatea" de informaţie secretă pe care o pot obţine grupurile neautorizate, sistemele de partajare a secretelor se clasifică în

- *Sisteme perfecte* de partajare: componentele deţinute de grupurile neautorizate nu oferă nici o informaţie (în sensul teooretic al termenului) despre secretul S;

- *Sisteme computaţional - sigure* de partajare: grupurile neautorizate au acces la o anumită cantitate de informaţie relativă la S, dar problema aflării secretului plecând de la această informaţie formează o problemă $\mathcal{NP}$ - completă.

Literatura de specialitate abundă în prezentarea de sisteme de partajare a secretelor. Acest capitol face o trecere în revistă a celor mai importante sisteme şi abordări legate de subiectul propus.

Vom începe cu o clasă de sisteme/scheme de partajare a secretelor, numite *sisteme majoritare.*[1]

4.1 Scheme de partajare majoritară

Definiţia 4.1. *Fie k, n $(2 \leq k \leq n)$ două numere întregi.*
O schemă de partajare (n, k) - majoritară este o metodă de partajare a unui secret S între membrii unei mulţimi $\mathcal{P}$= $\{P_1, \ldots, P_n\}$ de participanţi, astfel încât orice asociere de k participanţi să poată calcula S, lucru imposibil pentru asocieri de $k-1$ sau mai puţini participanţi.

Exemplele date la începutul acestui capitol sunt sistem $(3, 2)$ - majoritare.

Să formalizăm puţin aceste noţiuni:

Definiţia 4.2. *Se numeşte* **structură de acces** $\mathcal{A} \subseteq 2^{\mathcal{P}}$ *mulţimea*[2] *tuturor grupurilor "autorizate", care dispun de informaţia legală necesară construirii sistemului.*
Celelalte grupuri sunt numite "grupuri neautorizate".

Fie $B \in \mathcal{A}$ şi $B \subseteq C \subseteq \mathcal{P}$. Dacă mulţimea de participanţi C caută să determine secretul S, ea va reuşi lucrând numai cu participanţii din B şi ignorând participanţii din $C \setminus B$. Altfel spus, structura de acces $\mathcal{A}$ satisface condiţia de monotonie:

$$\text{Dacă } B \in \mathcal{A} \text{ şi } B \subseteq C \subseteq \mathcal{P}, \text{ atunci } C \in \mathcal{A}.$$

Vom presupune că orice structură de acces este *monotonă.*

Definiţia 4.3. *Fie $2 \leq k \leq n$. Structura*

$$\mathcal{A} = \{A \in 2^{\mathcal{P}} \mid card(A) \geq k\}$$

se numeşte structură de acces (n, k) **- majoritară**.

Deci o schemă de partajare (n, k) - majoritară este o structură monotonă de acces (n, k) - majoritară.

Un element $B \in \mathcal{A}$ este *minimal* dacă

$$(\forall A \in \mathcal{P})[A \subset B \implies A \notin \mathcal{A}]$$

Vom nota cu $\mathcal{A}_{min}$ mulţimea elementelor minimale autorizate din $\mathcal{A}$.

[1] *Threshold scheme* în engleză, *a seuil* în franceză.
[2] S-a notat cu $2^{\mathcal{P}}$ mulţimea tuturor submulţimilor lui $\mathcal{P}$.

Se observă că această mulţime – numită şi "bază autorizată de acces" – caracterizează complet $\mathcal{A}$. Mai exact,

$$\mathcal{A} = \{C \subseteq \mathcal{P} \mid \exists B \in \mathcal{A}_{min},\ B \subseteq C\}.$$

Spunem că $\mathcal{A}$ este *închiderea* lui $\mathcal{A}_{min}$ şi notăm prin $\mathcal{A}= \overline{\mathcal{A}_{min}}$.

Exemplul 4.1. *Fie* $\mathcal{P}= \{P_1, P_2, P_3, P_4\}$ *şi* $\mathcal{A}_{min} = \{\{P_1, P_2, P_4\}, \{P_1, P_3, P_4\}, \{P_2, P_3\}\}$.
Vom avea
$\mathcal{A}= \{\{P_1, P_2, P_4\}, \{P_1, P_3, P_4\}, \{P_2, P_3\}, \{P_1, P_2, P_3\}, \{P_2, P_3, P_4\}, \{P_1, P_2, P_3, P_4\}\}$.
Invers, fiind dat $\mathcal{A}$*, se vede imediat că* $\mathcal{A}_{min}$ *este mulţimea părţilor sale minimale.*

Referitor la grupurile neautorizate de participanţi, este evident că dacă o mulţime $B \subseteq \mathcal{P}$ este neautorizată, orice submulţime a sa va fi de asemenea neautorizată.
Deci se va lua ca bază de lucru mulţimea grupurilor maximale neautorizate.

Definiţia 4.4. *O mulţime* $B \in 2^{\mathcal{P}} \backslash \mathcal{A}$ *este maximal neautorizată dacă*

$$(\forall C \in 2^{\mathcal{P}})\ [B \subset C \quad \Longrightarrow \quad C \in \mathcal{A}].$$

Vom nota cu $\mathcal{NA}_{max}$ mulţimea mulţimilor maximale neautorizate. Atunci o structură de acces neautorizată $\mathcal{NA}= 2^{\mathcal{P}} \backslash \mathcal{A}$ va fi definită prin

$$\mathcal{NA} = \left\{A \in 2^{\mathcal{P}} \mid (\exists B \in \mathcal{NA}_{max})\ (A \subseteq B)\right\}$$

Exemplul 4.2. *Fie* $n = 4$ *şi structura de acces*
$\mathcal{A}= \{\{P_1, P_2\}, \{P_1, P_2, P_3\}, \{P_1, P_2, P_4\}, \{P_1, P_2, P_3, P_4\}, \{P_3, P_4\}, \{P_1, P_3, P_4\}, \{P_2, P_3, P_4\}\}$.
Atunci:
$\mathcal{A}_{min} = \{\{P_1, P_2\}, \{P_3, P_4\}\}$,
$\mathcal{NA}_{max} = \{\{P_1, P_3\}, \{P_1, P_4\}, \{P_2, P_3\}, \{P_2, P_4\}\}$,
$\mathcal{NA}= \{\emptyset, \{P_1\}, \{P_2\}, \{P_3\}, \{P_4\}, \{P_1, P_3\}, \{P_1, P_4\}, \{P_2, P_3\}, \{P_2, P_4\}\}$.

Vom nota cu S_i componenta din secretul S, cunoscută de participantul P_i.

Valoarea lui S este aleasă de un *arbitru* $D \notin \mathcal{P}$. D va distribui – printr-un canal securizat – componentele $S_1, \ldots S_n$ (ale secretului) membrilor grupului $\mathcal{P}$, astfel încât nici un participant P_i să nu cunoască componentele celorlalţi şi nici să fie capabil ca din S_i să poată recompune secretul S.

Ulterior, participanţii unei submulţimi $B \subseteq \mathcal{P}$ pot pune în comun componentele cunoscute de ei (sau să le dea unei autorităţi în care au încredere) cu scopul de a determina S. Ei trebuie să poată reuşi în această tentativă dacă şi numai dacă $card(B) \geq k$.

Să notăm cu $\mathcal{K}$ spaţiul tuturor secretelor posibile S şi cu $\mathcal{S}$ spaţiul componentelor (toate componentele S_i posibile ale unui secret S).

4.1.1 Schema lui Blakely

Propusă în 1979 ([7]), aceasta este – istoric – prima schemă de partajare a secretelor.

Fie q un număr prim, k şi n numere întregi pozitive ($k \leq n$) şi $\mathcal{K}= Z_q^k$, $\mathcal{S}= Z_q^{k+1}$. Dacă $S = (a_1, a_2, \ldots, a_k)$ este un secret, atunci schema lui Blakely este:

1. D alege $\alpha_{ij}, \beta_i \in Z_q$ $(1 \leq i \leq n,\ 1 \leq j \leq k)$ astfel ca

$$\begin{pmatrix} \alpha_{11} & \alpha_{12} & \ldots & \alpha_{1k} \\ \alpha_{21} & \alpha_{22} & \ldots & \alpha_{2k} \\ & & \vdots & \\ \alpha_{n1} & \alpha_{n2} & \ldots & \alpha_{nk} \end{pmatrix} \cdot \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_k \end{pmatrix} = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{pmatrix} \quad (mod\ q)$$

iar matricea $\begin{pmatrix} \alpha_{11} & \alpha_{12} & \ldots & \alpha_{1k} \\ \alpha_{21} & \alpha_{22} & \ldots & \alpha_{2k} \\ & & \vdots & \\ \alpha_{n1} & \alpha_{n2} & \ldots & \alpha_{nk} \end{pmatrix}$ să aibă rangul k.

2. Trimite fiecărui participant P_i componenta $S_i = (\alpha_{i1}, \alpha_{i2}, \ldots, \alpha_{ik}, \beta_i)$

Schema este o structură de acces (n, k) - majoritară deoarece:

- Fiecare participant P_i va construi – din componenta sa – ecuaţia diofantică

$$\alpha_{i1}x_1 + \ldots + \alpha_{ik}x_k = \beta_i \quad (mod\ q)$$

  care are printre soluţiile sale şi secretul $S = (a_1, \ldots, a_k)$.

- Orice grup de k parteneri va putea recompune secretul S rezolvând un sistem format din cele k ecuaţii liniare puse în comun.

Exemplul 4.3. *Fie $q = 31$ şi să considerăm o schemă $(3, 2)$ - majoritară, unde componentele participanţilor P_1, P_2, P_3 sunt:*

$$S_1 = (4, 29, 8),\ S_2 = (2, 1, 8),\ S_3 = (3, 27, 1)$$

Nici unul din participanţi nu poate afla singur secretul $S = (x, y) \in Z_{31} \times Z_{31}$.

Dacă se aliază însă P_1 cu P_3, ei au de rezolvat – în Z_{31} – sistemul liniar

$$\begin{cases} 4x + 29y = 8 \\ 3x + 27y = 1 \end{cases}$$

care are soluţia (unică) $S = (3, 2)$.

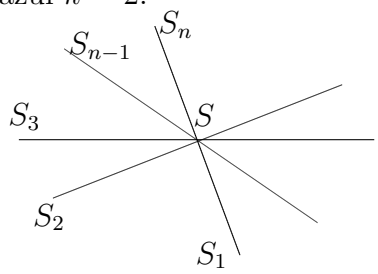
La acelaşi rezultat ajunge şi o alianţă P_1 cu P_2, P_2 cu P_3 sau P_1 cu P_2 şi P_3. Structura de acces este deci

$$\mathcal{A}= \{\{P_1, P_2\}, \{P_1, P_3\}, \{P_2, P_3\}, \{P_1, P_2, P_3\}\},$$

o structură $(3, 2)$ *- majoritară.*

Schema lui Blakely are şi o interpretare geometrică. Fiecare participant deţine ecuaţia unui hiperplan $(k-1)$ - dimensional, cu proprietatea că printre cele n hiperplanuri nu sunt două paralele. Intersecţia oricăror k astfel de hiperplanuri este un singur punct – totdeauna acelaşi – care constituie secretul S.

Figura următoare descrie cazul $k = 2$.



Schema lui Blakely nu constituie un sistem perfect, deoarece orice grup neautorizat ştie că secretul se află undeva la intersecţia hiperplanelor deţinute de membrii săi; deci grupul posedă o anumită cantitate de informaţie suplimentară, ceea ce reduce dimensiunea acestor hiperplane.

O securitate perfectă se atinge atunci când secretul S este numai una din coordonatele $a \in Z_q$ ale soluţiei $(a_1, \ldots, a_k)$.

4.1.2 Schema lui Shamir

Fie q $(q \geq n + 1)$ un număr prim şi $\mathcal{K}= Z_q$, $\mathcal{S} = Z_q$. Schema lui Shamir ([73]), definită în 1979, se bazează pe un polinom generat aleator $a(X)$ de grad cel mult $k-1$, în care termenul liber este S. Fiecare participant P_i cunoaşte un punct (x_i, y_i) de pe graficul acestui polinom.

1. D alege n elemente distincte $x_1, \ldots, x_n \in Z_q$ (x_i publice), fiecare x_i fiind comunicat lui P_i.

2. Dacă D intenţionează să repartizeze secretul $S \in Z_q$, el va genera $k-1$ elemente aleatoare $a_1, \ldots, a_{k-1} \in Z_q$ şi va construi polinomul
$$a(X) = S + \sum_{j=1}^{k-1} a_j X^j \quad (mod\ q).$$

3. D calculează $y_i = a(x_i)$ şi comunică această valoare lui P_i $(1 \leq i \leq n)$.

Fie acum o submulţime $\{P_{i_1}, \ldots, P_{i_k}\}$ de participanţi care doresc să reconstituie secretul. Ei ştiu valorile x_{i_j} şi $y_{i_j} = a(x_{i_j})$ pentru $1 \leq j \leq k$; $a(X) \in Z_q[X]$ este polinomul (secret) folosit de D. Cum gradul lui este cel mult $k-1$, putem scrie

$$a(X) = a_0 + a_1 X + \ldots + a_{k-1} X^{k-1}$$

unde $a_0, \ldots, a_{k-1} \in Z_q$ sunt necunoscute. Ele se află rezolvând sistemul liniar de k ecuaţii $y_{i_j} = a(x_{i_j}),\ \ 1 \leq j \leq k$.
Dacă ecuaţiile sunt independente, soluţia este unică, iar valoarea lui a_0 este chiar secretul S.

Exemplul 4.4. *Să presupunem* $q = 17,\ k = 3,\ n = 5$ *şi* $x_i = i\ (1 \leq i \leq 5)$.
Dacă mulţimea $B = \{P_1, P_3, P_5\}$ *de participanţi vrea să afle secretul, fiecare participant aducând informaţiile* 8, 10 *şi respectiv* 11, *ei vor scrie polinomul general* $a(X) = a_0 + a_1 X + a_2 X^2$ *şi vor reduce problema la rezolvarea în* Z_{17} *a sistemului liniar*

$$\begin{cases} a(1) = a_0 + a_1 + a_2 = 8 \\ a(3) = a_0 + 3a_1 + 9a_2 = 10 \\ a(5) = a_0 + 5a_1 + 8a_2 = 11 \end{cases}$$

Acesta admite soluţia unică în Z_{17} : $\quad a_0 = 13,\ a_1 = 10,\ a_2 = 2$.

Deci valoarea căutată este $S = 13$.

Teorema 4.1. *În schema de partajare Shamir, orice mulţime* B *de* k *participanţi poate reconstitui în mod unic secretul* S.

Demonstraţie. Fie $a(X) = a_0 + a_1 X + \ldots + a_{k-1} X^{k-1}$ polinomul ales de D, unde $a_0 = S$. Afirmaţia se reduce la a arăta că sistemul de ecuaţii $y_{i_j} = a(x_{i_j})\ \ (1 \leq j \leq k)$, de necunoscute $a_0, \ldots, a_{k-1}$, admite soluţie unică.
Determinantul acestui sistem este

$$\begin{vmatrix} 1 & x_{i_1} & x_{i_1}^2 & \ldots & x_{i_1}^{k-1} \\ 1 & x_{i_2} & x_{2_1}^2 & \ldots & x_{i_2}^{k-1} \\ & & \ldots & & \\ 1 & x_{i_k} & x_{i_k}^2 & \ldots & x_{i_k}^{k-1} \end{vmatrix} = \prod_{1 \leq j < t \leq k} (x_{i_t} - x_{i_j}) \quad (mod\ q)$$

Deoarece toate numerele x_i sunt distincte, iar Z_q este corp, rezultă că acest produs este nenul, deci sistemul are totdeauna soluţie unică, iar a_0 este chiar secretul căutat. □

Ce se întâmplă dacă un grup de $k-1$ participanţi încearcă să calculeze secretul S ?

Dacă procedează conform algoritmului anterior, vor obţine un sistem de $k-1$ ecuaţii cu k necunoscute. Fie y_0 o valoare arbitrară a lui S. Vom avea $y_0 = a_0 = a(0)$, care formează încă o ecuaţie a sistemului (în total sunt acum k ecuaţii). Acesta va avea de asemenea o soluţie unică.

Deci, pentru orice valoare $S \in Z_q$ există un polinom unic $a_S(X) \in Z_q[X]$ care verifică toate condiţiile:

$$y_0 = a_S(0), \qquad y_{i_j} = a_S(x_{i_j}), \quad 1 \leq j \leq k-1.$$

Rezultă că orice valoare a lui S este consistentă cu componentele deţinute de cei $k-1$ participanţi; asocierea lor nu oferă nici o informaţie suplimentară pentru aflarea secretului. Avem deci o structură de acces (n, k) - majoritară.

Mai există o modalitate de abordare a sistemului Shamir: folosind polinoamele de interpolare Lagrance.

În această interpretare, construcţia schemei de partajare (n, k) - majoritară este:

1. Se alege un număr prim $q > max\{S, n\}$.
2. Se defineşte polinomul

   $$a(X) = a_{k-1}X^{k-1} + \ldots + a_1X + a_0 \in Z_q[X]$$

   cu $a_0 = S$ şi a_i $(1 \leq i \leq k-1)$ arbitrari în Z_q.
3. Se determină componentele

   $$S_i = a(x_i), \quad 1 \leq i \leq n$$

   unde $x_1, x_2, \ldots, x_n \in Z_q$ sunt valori publice arbitrare, distincte două câte două. Aceste componente se trimit celor n participanţi.

Având componentele S_{i_j} $(1 \leq j \leq k)$ ale unui grup de acces, secretul S se poate obţine folosind formula de interpolare Lagrance. Fie polinomul

$$P(X) = \sum_{j=1}^{k} S_{i_j} \prod_{\substack{1 \leq t \leq k \\ t \neq j}} \frac{X - x_{i_t}}{x_{i_j} - x_{i_t}}.$$

Evident, acesta este un polinom de grad cel mult $k-1$, cu proprietatea $S_{i_j} = P(x_{i_j})$, $j = 1, \ldots, k$.
Cum un astfel de polinom este unic, rezultă că el este chiar polinomul $a(X)$.

Un grup B de k participanţi poate calcula $a(X)$ pe baza acestei formule. De fapt, nici nu este nevoie se determine tot polinomul: este suficient să obţină $S = a(0)$.

Deci, înlocuind în formulă pe X cu 0, avem

$$S = \sum_{j=1}^{k} S_{i_j} \prod_{\substack{1 \le t \le k \\ t \ne j}} \frac{x_{i_t}}{x_{i_t} - x_{i_j}}.$$

Dacă definim

$$b_j = \prod_{\substack{1 \le t \le k \\ t \ne j}} \frac{x_{i_t}}{x_{i_t} - x_{i_j}}, \qquad 1 \le j \le k$$

aceste valori pot fi precalculate şi făcute publice de către arbitru.

Secretul este atunci o combinaţie liniară de k componente:

$$S = \sum_{j=1}^{k} b_j S_{i_j}.$$

Exemplul 4.5. *Să considerăm o schemă de partajare Shamir* $(3, 2)$ *- majoritară.*

Fie $q = 31,\ x_1 = 20,\ x_2 = 6,\ x_3 = 11$ *componentele publice.*

Componentele secrete sunt $S_1 = 12,\ S_2 = 25$ *şi* $S_3 = 27$.

Să presupunem că mulţimea autorizată $\{P_2, P_3\}$ *doreşte să afle secretul* S.

Ea va calcula

$$S = S_2 \cdot \frac{x_3}{x_3 - x_2} + S_3 \cdot \frac{x_2}{x_2 - x_3} = \frac{25 \cdot 11 - 27 \cdot 6}{5} = 20 \cdot 25 = 4 \pmod{31}$$

La acelaşi rezultat se ajunge dacă se rezolvă în $Z_{31} \times Z_{31}$ *sistemul de ecuaţii liniare*

$$\begin{cases} 6a_1 + a_0 = 25 \\ 11a_1 + a_0 = 27 \end{cases}$$

cu soluţia $a_1 = 19,\ a_0 = 4$.

În subsidiar, se observă că sistemul Shamir a lucrat cu polinomul de interpolare $a(X) = 19X + 4$. *Într-adevăr,*

$S_1 = a(x_1) = a(20) = 12\ (mod\ 31)$,

$S_2 = a(x_2) = a(6) = 25\ (mod\ 31)$,

$S_3 = a(x_3) = a(11) = 27\ (mod\ 31)$.

Exemplul 4.6. *Să considerăm* $q = 11,\ n = 5$ *şi* $k = 3$.

Fie polinomul $a(X) = 2X^2 + 7X + 10 \in Z_{11}[X]$.

Secretul este $S = 10$, *iar componentele se calculează imediat (în* Z_{11}*):*

$S_1 = a(1) = 8,\ S_2 = a(2) = 10,\ S_3 = a(3) = 5,\ S_4 = a(4) = 4,\ S_5 = a(5) = 7$.

Pentru o mulţime autorizată $\{P_1, P_2, P_4\}$, *secretul poate fi reconstruit cu*

$$8 \cdot \frac{2}{2-1} \cdot \frac{4}{4-1} + 10 \cdot \frac{1}{1-2} \cdot \frac{4}{4-2} + 4 \cdot \frac{1}{1-4} \cdot \frac{2}{2-4}$$

Teorema 4.2. *Schema de partajare Shamir este perfectă.*

Demonstraţie. Dacă avem numai $k-1$ componente, sistemul de $k-1$ ecuaţii

$$\begin{cases} a_{k-1}x_{i_1}^{k-1} + \ldots + a_1 x_{i_1} &= S_{i_1} - a_0 \\ a_{k-1}x_{i_2}^{k-1} + \ldots + a_1 x_{i_2} &= S_{i_2} - a_0 \\ & \vdots \\ a_{k-1}x_{i_{k-1}}^{k-1} + \ldots + a_1 x_{i_{k-1}} &= S_{i_{k-1}} - a_0 \end{cases}$$

având necunoscutele $(a_{k-1}, \ldots, a_1)$ are soluţie unică pentru fiecare a_0. Deci este posibilă orice valoare a secretului S. □

Lema 4.1. *Dacă informaţia $grad(a(X)) = k-1 > 0$ este publică, atunci schema lui Shamir nu este perfectă.*

Demonstraţie. În acest caz, orice grup de $k-1$ utilizatori poate determina un element b_0, care în mod cert nu este a_0 (deci domeniul de valori $\mathcal{K}$ al secretului se micşorează).
Mai exact, folosind formula de interpolare Lagrance, se poate determina un polinom

$$Q(X) = b_{k-2}X^{k-2} + \ldots + b_1 X + b_0$$

cu proprietatea $Q(x_{i_j}) = S_{i_j} = a(x_{i_j})$ $(1 \leq j \leq k-1)$, care conduce la sistemul

$$\begin{cases} a_{k-1}x_{i_1}^{k-1} + (a_{k-2} - b_{k-2})x_{i_1}^{k-2} + \ldots + (a_1 - b_1)x_1 + (a_0 - b_0) &= 0 \\ & \vdots \\ a_{k-1}x_{i_{k-1}}^{k-1} + (a_{k-2} - b_{k-2})x_{i_{k-1}}^{k-2} + \ldots + (a_1 - b_1)x_{k-1} + (a_0 - b_0) &= 0 \end{cases}$$

Dacă presupunem $a_0 = b_0$, acest sistem de $k-1$ ecuaţii cu $k-1$ necunoscute $(a_{k-1}, \ldots, a_1)$ va avea soluţie unică:

$$a_{k-1} = 0, \quad a_{k-2} = b_{k-2}, \quad \ldots, \quad a_1 = b_1,$$

ceea ce contrazice presupunerea $a_{k-1} \neq 0$.

În concluzie, orice grup de $k-1$ participanţi poate determina un element b_0 care nu este secret; deci gradul lor de incertitudine nu va coincide cu cel de incertitudine al unui atacator extern. □

Observaţia 4.1.

1. *Mărimea fiecărei componente S_i nu depăşeşte mărimea secretului S (schema este "ideală").*

2. *Pentru o valoare fixată a lui k se pot adăuga sau elimina dinamic componente S_i (de exemplu, prin venirea sau ieşirea din sistem a noi participanţi) fără a afecta celelalte componente.*

3. *Componentele S_i pot fi modificate fără a schimba secretul S: singura schimbare constă în alegerea unui nou polinom $a(X)$ având S ca termen liber. Acest lucru se recomandă a se efectua periodic, pentru a păstra nivelul de securitate al sistemului.*

4. *McElliece şi Sarwate ([56]) au remarcat că schema Shamir are multe similitudini cu codurile Reed - Solomon; deci algoritmii de decodificare construiţi pentru aceste coduri pot fi utilizaţi pentru generalizarea schemei de partajare Shamir.*

Pentru cazul $n = k$ se poate construi o variantă simplificată a algoritmului Shamir. Ea funcţionează pentru $\mathcal{K} = Z_m$, $\mathcal{S} = Z_m$ (m nu este obligatoriu număr prim şi – chiar mai mult – este posibil ca $m \leq n$). Noul algoritm este:

1. D alege aleator $n-1$ elemente $y_1, \ldots, y_{n-1} \in Z_m$;
2. D calculează $y_n = S - \sum_{i=1}^{n-1} y_i \pmod m$;
3. Fiecare element y_i este transmis (prin canal securizat) lui P_i, $i = 1, \ldots, n$.

Cei n participanţi pot determina secretul S pe baza formulei

$$S = \sum_{i=1}^{n} y_i \pmod m.$$

Evident, $n-1$ participanţi nu-l pot obţine pe S. Chiar dacă pun în comun componentele lor, ei pot determina valoarea $S - y$, unde y este componenta celui care lipseşte. Cum y este o valoare aleatoare din Z_m, nu se va obţine nici o informaţie suplimentară referitoare la secret. Acesta este deci o structură de acces (n, n) - majoritară.

4.1.3 Schema Mignotte

Schema de partajare (n, k) - majoritară a lui Mignotte ([59]) se bazează pe secvenţe de numere întregi numite *şiruri Mignotte*.

Definiţia 4.5. *Fie n ($n \geq 2$) un număr întreg şi $2 \leq k \leq n$. Un şir (n, k) - Mignotte este o secvenţă de numere întregi pozitive $p_1 < p_2 < \ldots < p_n$ prime două câte două, cu proprietatea*

$$\prod_{i=0}^{k-2} p_{n-i} < \prod_{i=1}^{k} p_i.$$

Această relaţie este echivalentă cu

$$\max_{1 \leq i_1 < \ldots < i_{k-1} \leq n} (p_{i_1} \ldots p_{i_{k-1}}) < \min_{1 \leq i_1 < \ldots < i_k \leq n} (p_{i_1} \ldots p_{i_k})$$

Exemplul 4.7. *Şirul*

$$5, 7, 9, 11, 13$$

formează o secvenţă $(5, 3)$ *- Mignotte. Cele cinci numere sunt prime (deci şi prime între ele), iar inegalitatea din Definiţia 4.5 este* $p_4 \cdot p_5 < p_1 \cdot p_2 \cdot p_3$*, verificată evident.*

Schema de partajare (n, k) - majoritară Mignote se defineşte astfel:

1. D alege un şir (n, k) - Mignotte şi calculează
$$\alpha = \prod_{i=1}^{k} p_i, \qquad \beta = \prod_{i=0}^{k-2} p_{n-i}.$$
2. D alege $S \in (\beta, \alpha)$ (în general, S este generat aleator).
3. D calculează $S_i = S \pmod{p_i}$ şi trimite fiecărui utilizator P_i perechea $(p_i, S_i), \quad i = 1, \ldots, n.$

Fiind cunoscute k componente distincte $S_{i_1}, \ldots, S_{i_k}$, secretul S poate fi aflat pe baza Teoremei Chineze a Restului (TCR), fiind soluţia unică modulo $p_{i_1} p_{i_2} \ldots p_{i_k}$ a sistemului de congruenţe

$$\begin{cases} x \equiv S_{i_1} \pmod{p_{i_1}} \\ \quad \vdots \\ x \equiv S_{i_k} \pmod{p_{i_k}} \end{cases}$$

Pentru a asigura un ordin de securitate acceptabil, trebuie folosit un şir (n, k) - Mignotte cu o valoare $(\alpha - \beta)/\beta$ mare (o metodă de generare a unor astfel de şiruri este prezentată în [48], pagina 9).

Exemplul 4.8. *Folosind şirul Mignotte din Exemplul 4.7 se determină*

$$\alpha = 5 \cdot 7 \cdot 9 = 315, \qquad \beta = 11 \cdot 13 = 143.$$

Să considerăm secretul $285 \in (143, 315)$.

Cele cinci componente sunt

$$\begin{array}{lll} S_1 = S \pmod 5 = 0, & S_2 = S \pmod 7 = 5, & S_3 = S \pmod 9 = 6, \\ S_4 = S \pmod{11} = 10, & S_5 = S \pmod{13} = 12 & \end{array}.$$

Pentru grupul autorizat $\{P_1, P_3, P_4\}$ *trebuie rezolvat sistemul*

$$\begin{cases} x \equiv 0 \pmod 5 \\ x \equiv 6 \pmod 9 \\ x \equiv 10 \pmod{11} \end{cases}$$

a cărui soluţie unică este 285.

De remarcat că – deoarece $\frac{\beta-\alpha}{\beta} = 1.2$ *– în intervalul* (β, α) *există puţine numere care pot fi luate drept secret* S *accesibil oricărui grup autorizat.*

De exemplu, pentru $S = 300$*, participanţii* P_3 *şi* P_4 *posedă aceeaşi componentă:* $S_3 = S_4 = 3$*; deci nu există nici un grup autorizat de forma* $\{P_3, P_4, x\}$ *care să aibă acces la secretul* S*.*

Evident, schema Mignotte nu este perfectă; avantajul ei este însă acela că oferă componente mici, şi deci poate fi utilizată în aplicaţii în care un factor important de lucru constă în compactificarea componentelor.

Sorin Iftene ([41]) extinde schema de partajare majoritară Mignotte, folosind şirurile Mignotte generalizate (elementele sale nu mai sunt obligatoriu prime două câte două).

Schema Asmuth-Bloom

Schema propusă de Asmuth and Bloom ([1]) este foarte asemănătoare cu schema Mignotte. Ea foloseşte un şir de numere întregi pozitive prime două câte două

$$p_0,\ p_1 < \ldots < p_n$$

cu proprietatea

$$p_0 \cdot \prod_{i=0}^{k-2} p_{n-i} < \prod_{i=1}^{k} p_i.$$

Fiind dat un şir Asmuth-Bloom, schema de partajare (n, k) - majoritară este definită astfel:

1. D stabileşte secretul $S \in Z_{p_0}$.
2. Componentele S_i $(1 \leq i \leq n)$ sunt definite $S_i = (S + r \cdot p_0)\ (mod\ p_i)$, unde r este un întreg arbitrar cu proprietatea $S + r \cdot p_0 \in Z_{p_1 \ldots p_k}$.
3. D trimite componentele S_i $(1 \leq i \leq n)$ celor n participanţi.

Fiind date k componente distincte $S_{i_1}, \ldots, S_{i_k}$, secretul S se obţine prin $S = x_0\ (mod\ p_0)$, unde x_0 este soluţia modulo $p_{i_1} \ldots p_{i_k}$ (obţinută cu TCR) a sistemului

$$\begin{cases} x \equiv S_{i_1} & (mod\ p_{i_1}), \\ \quad \vdots & \\ x \equiv S_{i_k} & (mod\ p_{i_k}) \end{cases}$$

Exemplul 4.9. *Să considerăm şirul Asmuth - Bloom*

$$p_0 = 5,\ p_1 = 11,\ p_2 = 13,\ p_3 = 17,\ p_4 = 19,\ p_5 = 23$$

Evident, inegalitatea $p_0 \cdot p_4 \cdot p_5 < p_1 \cdot p_2 \cdot p_3$ *este verificată (revine la* $2185 < 2431$*).*
Pe baza acestui şir definim schema de partajare $(5, 3)$ *- majoritară Asmuth-Bloom:*
Fie $S = 2 \in Z_5$*. Alegem* $r = 317$ *şi avem* $S + r \cdot p_0 = 1587$*.*
Cele cinci componente sunt:

$$S_1 = 1587 \ (mod\ 11) = 3, \quad S_2 = 1587 \ (mod\ 13) = 1, \quad S_3 = 1587 \ (mod\ 17) = 6$$
$$S_4 = 1587 \ (mod\ 19) = 10, \quad S_5 = 1587 \ (mod\ 23) = 0.$$

Să considerăm mulţimea autorizată $\{P_2, P_3, P_4\}$*.*
Cei trei patricipanţi vor avea de rezolvat sistemul de congruenţe

$$x \equiv 1 \ (mod\ 13), \qquad x \equiv 6 \ (mod\ 17), \qquad x \equiv 10 \ (mod\ 19)$$

care are soluţia (unică în Z_{4199}*)*

$$x_0 = 313 \cdot 6 \cdot 1 + 247 \cdot 2 \cdot 6 + 221 \cdot 8 \cdot 10 = 22582 \ (mod\ 4199) = 1587$$

Se poate determina acum secretul $S = 1587 \ (mod\ 5) = 2$*.*

Şirul Asmuth-Bloom poate fi generalizat eliminând condiţia de numere prime între ele. Practic, se poate utiliza orice şir $p_0, p_1, \ldots, p_n$ care verifică inegalitatea[3]

$$p_0 \cdot \max_{1 \le i_1 < i_2 < \ldots < i_{k-1} \le n} \left([p_{i_1}, \ldots, p_{i_{k-1}}] \right) < \min_{1 \le i_1 < i_2 < \ldots < i_k \le n} \left([p_{i_1}, \ldots, p_{i_k}] \right)$$

Este uşor de remarcat că dacă se înmulţesc elementele p_i $(i > 0)$ ale unui şir Asmuth-Bloom $p_0, p_1, \ldots, p_n$ cu o valoare fixată $r \in \mathcal{Z}$, $(r, p_0 \ldots p_n) = 1$, se obţine un şir Asmuth-Bloom generalizat.

4.1.4 Scheme de partajare majoritar ponderate

Într-o schemă de partajare *majoritar ponderată*, fiecărui utilizator i se asociază un număr pozitiv (numit "pondere"); secretul poate fi reconstruit dacă şi numai dacă suma ponderilor participanţilor este cel puţin egală cu o valoare limită fixată.

Shamir ([73]) este primul care defineşte astfel de scheme, prezentând scenariul unei companii, în care secretul poate fi acoperit de 3 directori, de doi directori şi un vicepreşedinte, sau de către preşedinte. Ideea de bază este de a acorda mai multe componente utilizatorilor mai importanţi (aici preşedintele primeşte 3 componente, fiecare vice-preşedinte are câte două componente, iar un director deţine numai o componentă a secretului).

Structurile de acces majoritar ponderate se definesc astfel:

[3]Notaţia $[x, y]$ semnifică cel mai mic multiplu comun al numerelor întregi pozitive x şi y.

Definiţia 4.6. *Fie* $n \geq 2$, $x = (x_1, \ldots, x_n)$ *un vector de numere întregi pozitive şi numărul întreg* $w \in \left(2, \sum_{i=1}^{n} x_i\right)$. *Structura de acces*

$$\mathcal{A} = \left\{ A \in 2^{\mathcal{P}} \mid \sum_{P_i \in A} x_i \geq w \right\}$$

se numeşte structură (x, w, n) *- majoritar ponderată.*

Într-o astfel de schemă, un grup $\{P_{i_1}, \ldots, P_{i_t}\}$ este autorizat dacă şi numai dacă $\{i_1, \ldots, i_t\}$ este o mulţime de acces într-o structură (x, w, n) - majoritar ponderată: $\sum_{j=1}^{t} x_{i_j} \geq w$.

Parametrii $x_1, \ldots, x_n$ se numesc *ponderi* iar w este *limita* schemei de partajare.
Dacă $\mathcal{A}$ este o structură de acces (x, w, n) - majoritar ponderată, orice sistem de partajare construit pe baza ei se numeşte *schemă de partajare a secretelor* (x, w, n) *- majoritar ponderată.*

Observaţia 4.2. *O schemă de partajare a secretelor* (n, k) *- majoritară este un caz particular de schemă* (x, w, n) *- majoritar ponderată cu* $x_1 = \ldots = x_n = 1$ *şi* $w = k$.

Benaloh şi Leichter ([5]) au demonstrat că există structuri de acces care nu sunt majoritar ponderate.

Exemplul 4.10. *Fie* $n = 4$ *şi* $\mathcal{A}_{min} = \{\{P_1, P_2\}, \{P_3, P_4\}\}$ *(a se vedea şi Exemplul 4.2). Să presupunem că lucrăm cu o structură de acces majoritar ponderată, cu ponderile* x_1, x_2, x_3, x_4 *şi limita* w. *Deci*

$$x_1 + x_2 \geq w, \quad x_3 + x_4 \geq w.$$

Adunând aceste inegalităţi, obţinem $x_1 + x_2 + x_3 + x_4 \geq 2w$, *deci*

$$2 \cdot max(x_1, x_2) + 2 \cdot max(x_3, x_4) \geq 2w$$

sau $max(x_1, x_2) + max(x_3, x_4) \geq w$.
În concluzie, una din mulţimile $\{P_1, P_3\}, \{P_1, P_4\}, \{P_2, P_3\}, \{P_2, P_4\}$ *este o mulţime autorizată de acces, dar nu este generată de baza* $\mathcal{A}_{min}$.

Cea mai simplă metodă de construcţie a unei scheme (x, w, n) - majoritar ponderate constă în utilizarea unei scheme de partajare (N, w) - majoritare, în care $N = \sum_{i=1}^{n} x_i$.

Detaliind, fie $s_1, \ldots, s_N$ componentele corespunzătoare unui secret S în raport cu o schemă arbitrară de partajare a secretelor (N, w) - majoritară. Considerăm o partiţie oarecare $\{X_1, \ldots, X_n\}$ a mulţimii $\{1, 2, \ldots, N\}$, cu $card(X_i) = x_i, (1 \leq i \leq n)$.

Definim atunci componentele structurii de acces majoritar ponderate prin

$$S_i = \{s_j \mid j \in X_i\}, \quad i = 1 \ldots n$$

Scheme majoritar ponderate bazate pe TCR

O extensie (naturală) a şirului Mignotte generalizat este:

Definiţia 4.7. *Fie $x = (x_1, \ldots, x_n)$ $(n \geq 2)$ un şir de ponderi şi w o limită. Un şir (x, w, n) - Mignotte este un şir $p_1, \ldots, p_n$ de numere întregi pozitive cu proprietatea*

$$\max_{\substack{A \in 2^{\mathcal{P}} \\ \sum_{P_i \in A} x_i \leq w-1}} ([\{p_i \mid P_i \in A\}]) < \min_{\substack{A \in 2^{\mathcal{P}} \\ \sum_{P_i \in A} x_i \geq w}} ([\{p_i \mid P_i \in A\}])$$

Observaţia 4.3. *Pentru $x_1 = \ldots = x_n = 1$ şi $w = k$, un şir $p_1, \ldots, p_n$ este şir (x, w, n) - Mignotte dacă şi numai dacă $p_1, \ldots, p_n$ este un şir (n, k) - Mignotte generalizat.*

În aceleaşi ipoteze, un şir $p_1, \ldots, p_n$ cu elemente prime între ele, este un şir $(x, , w, n)$ - Mignotte dacă şi numai dacă $p_1, \ldots, p_n$ este un şir (n, k) - Mignotte (conform Definiţiei 4.5).

O modalitate de construcţie a şirurilor (x, w, n) - Mignotte este:

Fie $p'_1, \ldots, p'_N$ un şir (N, w) - Mignotte generalizat, unde $N = \sum_{i=1}^{n} x_i$.

Definim $p_i = \left[\{p'_j \mid j \in X_i\}\right]$, $(1 \leq i \leq n)$, unde $\{X_1, \ldots, X_n\}$ este o partiţie arbitrară a mulţimii $\{1, 2, \ldots, N\}$ astfel încât $card(X_i) = x_i$ $(1 \leq i \leq n)$.

Se obţine

$$\max_{A \in T} ([\{p_i \mid P_i \in A\}]) = \max_{A \in T} \left([\{[\{p'_j \mid j \in X_i\}] \mid P_i \in A\}]\right) = \max_{A \in T} \left([\{p'_j \mid j \in \bigcup_{P_i \in A} X_i\}]\right)$$

unde am notat $T = \{A \in 2^{\mathcal{P}} \mid \sum_{P_i \in A} x_i \leq w - 1\}$.

În plus, pentru orice mulţime $A \in 2^{\mathcal{P}}$ cu $\sum_{P_i \in A} x_i \leq w - 1$ se obţine

$$card\left(\{p'_j \mid j \in \bigcup_{P_i \in A} X_i\}\right) = \sum_{P_i \in A} card(X_i) = \sum_{P_i \in A} x_i \leq w - 1, \text{ şi deci}$$

$$\max_{A \in T} ([\{p_i \mid P_i \in A\}]) \leq \max_{1 \leq i_1 < \ldots < i_{w-1} \leq N} \left([\{p'_{i_1}, \ldots, p'_{i_{w-1}}\}]\right).$$

Printr-un raţionament similar se arată şi relaţia

$$\min_{1 \leq i_1 < \ldots < i_w \leq N} \left([\{p'_{i_1}, \ldots, p'_{i_w}\}]\right) \leq \min_{A \in U} ([\{p_i \mid P_i \in A\}])$$

unde s-a notat $U = \{A \in 2^{\mathcal{P}} \mid \sum_{P_i \in A} x_i \geq w\}$.

Cu aceste două relaţii, din faptul că $p'_1, \ldots, p'_N$ este un şir (N, w) - Mignotte generalizat şi din Definiţia 4.7, rezultă că $p_1, \ldots, p_n$ este un şir (x, w, n) - Mignotte.

Exemplul 4.11. *Fie* $n = 4$, *ponderile* $x_1 = x_2 = 1$, $x_3 = x_4 = 2$ *şi limita* $w = 3$. *Se obţine imediat* $N = 6$.

Folosim şirul $7, 11, 13, 17, 19, 23$ *(care este un şir* $(3, 6)$ *- Mignotte generalizat). Considerând partiţia* $\{\{6\}, \{5\}, \{1, 4\}, \{2, 3\}\}$ *a mulţimii* $\{1, 2, 3, 4, 5, 6\}$, *va rezulta şirul*

$$23, 19, 119, 143 \quad \textit{unde} \quad 119 = [7, 17],\ 143 = [11, 13]$$

ca un şir $((1, 1, 2, 2), 3, 4)$ *- Mignotte.*

Pe baza unui şir (x, w, n) - Mignotte $p_1, \ldots, p_n$, se poate construi o schemă de partajare (x, w, n) - majoritar ponderată, în felul următor:

1. D calculează U şi
$$\alpha = \min_{A \in U} \left([\{p_i \mid P_i \in A\}] \right), \qquad \beta = \max_{A \in T} \left([\{p_i \mid P_i \in A\}] \right);$$
2. D generează (aleator) secretul $S \in [\beta + 1, \alpha - 1]$.
3. Componentele sunt $S_i = S \ (mod\ p_i), \quad (1 \leq i \leq n)$.
4. Fiecare participant P_i $(1 \leq i \leq n)$ primeşte de la D perechea (S_i, p_i).

Pentru o mulţime de componente $\{S_i \mid P_i \in A\}$, unde mulţimea $A \in \mathcal{A}$ verifică inegalitatea $\sum_{i \in A} x_i \geq w$, secretul S poate fi obţinut ca soluţie (unică) modulo $[\{p_i \mid P_i \in A\}]$ a sistemului

$$\{\ x \equiv S_i \ (mod\ p_i), \quad P_i \in A\ .$$

Securitatea schemei (x, w, n) **- majoritar ponderată**: Pentru un set de componente $\{S_i \mid P_i \in A\}$, unde A verifică inegalitatea $\sum_{P_i \in A} x_i \leq w - 1$, singura informaţie care se poate obţine prin aflarea soluţiei $x_0 \in Z_{[\{p_i \mid P_i \in A\}]}$ a sistemului

$$\{\ x \equiv S_i \ (mod\ p_i), \quad P_i \in A$$

este $S \equiv x_0 \ (mod\ [\{p_i \mid P_i \in A\}])$.

Într-adevăr, prin alegerea secretului S $(S > \beta)$, vom avea $S \notin Z_{[\{p_i \mid P_i \in A\}]}$, deci acesta nu va fi soluţia unică modulo $[\{p_i \mid P_i \in A\}]$ a sistemului de mai sus.

Alegând şiruri (x, w, n) - Mignotte cu valori mari pentru $\dfrac{\alpha - \beta}{\beta}$, problema găsirii secretului S, ştiind că este în intervalul $[\beta + 1, \alpha - 1]$ şi $S \equiv x_0 \ (mod\ [\{p_i \mid P_i \in A\}])$, pentru o mulţime neautorizată A, este $\mathcal{NP}$ - completă.

Exemplul 4.12. *Fie* $n = 4$, *ponderile* $x_1 = x_2 = 1$, $x_3 = x_4 = 2$ *şi limita* $w = 3$. *Structura de acces majoritar ponderată este generată de* $\mathcal{A}_{min} = \{\{1,3\},\{1,4\},\{2,3\},\{2,4\},\{3,4\}\}$, *iar* $\mathcal{A}_{max} = \{\{1,2\},\{3\},\{4\}\}$.

Conform Exemplului 4.11, şirul $23, 19, 119, 143$ *este un şir* $((1,1,2,2),3,4)$ *- Mignotte. Calculăm*

$\alpha = min([23,119],[23,143],[19,119],[19,143],[119,143]) = 2261$ *respectiv*

$\beta = max([23,19],119,143) = 437$.

O schemă de partajare a secretelor $((1,1,2,2),3,4)$ *- ponderat majoritară este:*

- $S \in [438, 2260]$ *este generat aleator; de exemplu, fie* $S = 601$.
- *Componentele sunt*

$$S_1 = 601 \ (mod\ 23) = 3,$$
$$S_2 = 601 \ (mod\ 19) = 12,$$
$$S_3 = 601 \ (mod\ 119) = 6,$$
$$S_4 = 601 \ (mod\ 143) = 29.$$

Considerând de exemplu componentele $S_1 = 3$ *şi* $S_3 = 6$ *(puse în comun de mulţimea autorizată de acces* $\{P_1, P_3\}$*), secretul* S *poate fi obţinut ca soluţie în* Z_{2737} *a sistemului*

$$\begin{cases} x \equiv 3 & (mod\ 23) \\ x \equiv 6 & (mod\ 119) \end{cases}$$

care este 601.

În schimb $A = \{P_1, P_2\}$ *nu corespunde unei mulţimi autorizate de acces. Într-adevăr, din componentele* $S_1 = 3$, $S_2 = 12$, *secretul* S *nu poate fi obţinut din soluţia în* Z_{437} *a sistemului*

$$\begin{cases} x \equiv 3 & (mod\ 23) \\ x \equiv 12 & (mod\ 119) \end{cases}$$

Se verifică imediat că acest sistem are soluţia unică 164.

4.1.5 Schemă majoritară bazată pe dispersia informaţiei

Ideea de schemă majoritară de dispersie a informaţiei a fost introdusă de Rabin ([67]).

Definiţia 4.8. *Se dau numerele întregi* n, k $(2 \leq k \leq n)$. *O schemă de dispersie a informaţiei* (n,k) *- majoritară este o metodă de generare* $(S, (F_1, \ldots, F_n))$ *cu proprietatea că pentru orice mulţime* $A \in 2^{\mathcal{P}}$ *cu* $card(A) = k$, *problema aflării elementului* S *din mulţimea* $\{F_i \mid P_i \in A\}$, *este "uşoară".*

Elementul S este numit "informaţia" iar $F_1, \ldots, F_n$ sunt numite "fragmentele lui S".

Singura diferenţă dintre noţiunea de dispersie a informaţiei şi cea de partajare a secretelor constă în faptul că în primul caz nu există nici o restricţie referitoare la grupurile neautorizate.

Krawczyk ([49]) a definit o schemă de dispersie a informaţiei (n, k) - majoritară, foarte apropiată de schema de partajare a lui Shamir:

- Fie informaţia $S = (a_0, \ldots, a_{k-1})$ cu elemente dintr-un corp finit. Se defineşte polinomul $a(X) = a_0 + a_1X + \ldots + a_{k-1}X^{k-1}$.
- Se construiesc fragmentele $F_1, \ldots, F_n$ prin $F_i = a(x_i)$ $(1 \leq i \leq n)$, unde $x_1, \ldots, x_n$ sunt valori publice distincte, generate aleator.
- Fiind dat un grup $A \in 2^{\mathcal{P}}$ cu $card(A) = k$ şi fragmentele $\{F_i \mid P_i \in A\}$, polinomul $a(X)$ (şi deci informaţia S) se poate obţine cu formula de interpolare Lagrange:

$$\sum_{P_i \in A} \left(F_i \cdot \prod_{P_j \in A \setminus \{P_i\}} \frac{X - x_j}{x_i - x_j} \right).$$

Diferenţa între această schemă şi schema de partajare majoritară a lui Shamir constă în faptul că aici informaţia este reprezentată printr-un polinom complet, pe când la Shamir ea era conţinută doar în termenul liber al unui polinom.

Pe baza acestei scheme, Krawczyk a definit un protocol majoritar de partajare a secretelor, computaţional sigur. O formă a sa este:

1. D face public un algoritm de criptare e.
2. D alege aleator o cheie K şi calculează $\overline{S} = e_K(S)$.
3. D foloseşte o schemă de dispersie a informaţiei (n, k) - majoritară pentru a "sparge" secretul $\overline{S}$ în n fragmente $F_1, \ldots, F_n$.
4. D foloseşte o schemă perfectă de partajare a secretelor (n, k) - majoritară pentru a construi componentele $K_1, \ldots, K_n$ corespunzătoare cheii secrete K.
5. Se definesc componentele $S_i = (F_i, K_i)$ $i = 1 \ldots n$, care se distribuie utilizatorilor prin canale securizate.

Fiind date k componente distincte $S_{i_1} = (F_{i_1}, K_{i_1}), \ldots, S_{i_k} = (F_{i_k}, K_{i_k})$, secretul S poate fi recompus astfel:

1. Se determină $\overline{S}$ folosind algoritmul de reconstrucţie aplicat lui $F_{i_1}, \ldots, F_{i_k}$.
2. Se determină cheia K folosind algoritmul de reconstrucţie pentru $K_{i_1}, \ldots, K_{i_k}$.
3. Se calculează secretul $S = d_K(\overline{S})$.

Observaţia 4.4. *Referitor la cantitatea de informaţie deţinută de un participant: Lungimea celei de a i-a componente este $|F_i| + |K_i|$; deci ea depinde atât de schema de dispersie a informaţiei cât şi de schema de partajare folosită.*
Dacă se foloseşte un sistem de criptare care păstrează lungimea ($\forall K \forall x,\ |e_K(x)| = |x|$), o schemă ideală de dispersie majoritară a informaţiei $\left(|F_i| = \frac{|\overline{S}|}{k},\ \forall i = 1, \ldots, n\right)$ *şi o schemă ideală de partajare a secretelor, atunci fiecare componentă S_i va avea lungimea* $\frac{|S|}{k} + |K|$.

4.2 Scheme de partajare unanime

În cazul $\mathcal{A} = \mathcal{A}_{min} = \{P_1, P_2, \ldots, P_n\}$, o $\mathcal{A}$ - schemă de partajare a secretelor se numeşte *unanimă*[4] de ordin n.
Pentru astfel de scheme, secretul este aflat numai prin participarea tuturor utilizatorilor implicaţi.

Evident, o schemă de partajare unanimă de ordin n este echivalentă cu o schemă de partajare (n, n) - majoritară şi – reciproc – orice schemă de partajare a secretelor (n, n) - majoritară poate fi utilizată în realizarea unei scheme de partajare unanime.

O schemă de partajare unanimă foarte simplă este propusă de Karnin, Greene şi Hellman ([44]):

1. Secretul S este un număr aleator din Z_q ($q > 2$ număr arbitrar fixat).
2. D generează aleator componentele $S_i \in Z_q,\ (1 \leq i \leq n-1)$.
   După aceea determină $S_n = S - \sum_{i=1}^{n-1} S_i \ (mod\ q)$.
3. D trimite fiecărui participant P_i componenta $S_i \quad i = 1, \ldots, n$

Secretul S poate fi reconstruit cu relaţia $S = \sum_{i=1}^{n} S_i \ (mod\ q)$.

Exemplul 4.13. *Să considerăm $n = 20,\ q = 15$, şi fie secretul $S = 4 \in Z_{15}$.*
Dacă se definesc componentele $S_i = i$ $(1 \leq i \leq 19)$, ultima componentă va fi

$$S_{20} = 4 - \sum_{i=1}^{19} i = 4 - 190 = -186 = 9 \ (mod\ 15)$$

[4]în engleză "unanimous consent secret sharing scheme".

Secretul S poate fi recompus numai prin însumarea celor 20 componente:

$$S = \sum_{i=1}^{20} S_i = 1 + 2 + \ldots + 19 + 9 = 199 = 4 \ (mod\ 15)$$

4.3 Scheme bazate pe grafuri pentru structuri de acces

O structură de acces în care orice mulţime minimală de acces are două elemente se numeşte *structură de acces 2 - omogenă* sau *structură de acces bazată pe grafuri* (deoarece grupurile minimale de acces pot fi specificate în acest caz prin arcele unui graf).

Definiţia 4.9. *Un graf $G = (V, E)$ este multipartit complet dacă V se poate partiţiona în submulţimile $V_1, \ldots, V_s$ astfel încât $\{x, y\} \in E$ dacă şi numai dacă $x \in V_i$, $y \in V_j$ cu $i \neq j$. Mulţimile V_i se numesc componente.*
Dacă $card(V_i) = n_i$ $(1 \leq i \leq s)$, graful este notat $K_{n_1,\ldots,n_s}$.
Graful multipartit complet $K_{1,\ldots,1}$ cu s componente este de fapt un graf complet şi se notează K_s.

Teorema 4.3. *Fie G un graf conex. Există o schemă ideală de partajare a secretelor pentru structura de acces specificată de G dacă şi numai dacă G este un graf multipartit complet.*

Stinson ([77]) construieşte o schemă ideală de partajare a secretelor, bazată pe structura de acces specificată de graful $K_{n_1,n_2,\ldots,n_s} = (V, E)$:

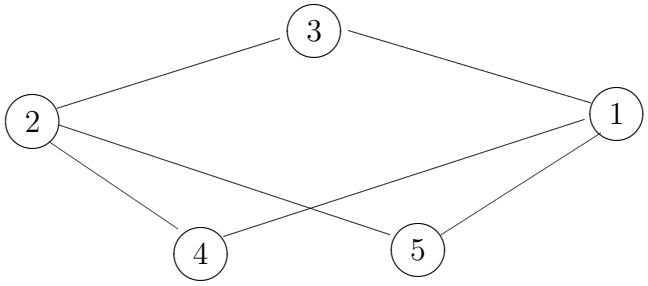
1. Fie $q > m = card(V)$ un număr prim şi $V_1, \ldots, V_s$ componentele grafului $K_{n_1,n_2,\ldots,n_s}$ (având nodurile numerotate $1, 2, 3, \ldots, m$).
2. D generează s numere aleatoare distincte $x_1, \ldots, x_s \in Z_q$.
3. Dacă $S \in Z_q$ este un secret, componentele sale se definesc prin
$$S_i = x_j \cdot S + r \ (mod\ q)$$
pentru $i \in V_j$ $(1 \leq j \leq s)$ şi $r \in Z_q$ arbitrar fixat.
4. D trimite fiecărui participant $u_i \in V_j$ componenta (S_i, x_j), $1 \leq j \leq s$.

Deci $\mathcal{K} = Z_q$, $\mathcal{S} = Z_q \times Z_q$.
Oricare doi utilizatori $(u_1, u_2) \in V_{j_1}, \times V_{j_2}$ $(j_1 \neq j_2)$ pot recompune secretul S după

formula

$$S = \frac{S_{u_1} - S_{u_2}}{x_{j_1} - x_{j_2}} \quad (mod\ q).$$

Exemplul 4.14. *Să considerăm graful multipartit complet* $K_{2,3}$*:*



Avem $s = 2$ *şi* $V_1 = \{P_1, P_2\},\ V_2 = \{P_3, P_4, P_5\}$.

Fie $q = 11$ *şi să alegem aleator valorile* $x_1 = 7,\ x_2 = 4,\ r = 8$.

Pentru secretul $S = 10$*, componentele sale sunt:*

$$S_1 = S_2 = x_1 \cdot S + r = 7 \cdot 10 + 8\ (mod\ 11) = 1,$$

$$S_3 = S_4 = S_5 = x_2 \cdot S + r = 4 \cdot 10 + 8\ (mod\ 11) = 4.$$

Dacă participanţii P_2 *şi* P_3 *vor să recompună secretul, ei vor calcula*

$$S = \frac{S_1 - S_4}{x_1 - x_2} = \frac{1-4}{7-4} = (-3) \cdot 3^{-1} = -1 = 10\ (mod\ 11)$$

4.4 Construcţia circuitelor monotone

Ideea ([5]) constă în construirea unui circuit combinaţional care "recunoaşte" structura de acces şi generează un sistem de partajare a secretului. Un astfel de circuit este numit de autori (Benaloh şi Leichter) *circuit monoton*.

Fie $\mathbf{C}$ un circuit combinaţional[5] cu n intrări notate prin variabilele booleene $x_1, \ldots, x_n$ (corespunzătoare celor n participanţi $P_1, \ldots, P_n$) şi o ieşire booleană $y = \mathbf{C}(x_1, \ldots, x_n)$. Presupunem că la construcţia circuitului sunt folosite numai porţi AND şi OR (fără porţi NOT). Un astfel de circuit este numit "monoton" dacă modificarea valorii unei intrări din 0 în 1 nu va implica niciodată transformarea ieşirii y din 1 în 0.

Vom nota

$$B(x_1, \ldots, x_n) = \{P_i \mid x_i = 1\}$$

mulţimea participanţilor asociaţi în mulţimea B. Presupunând că circuitul $\mathbf{C}$ este monoton, vom avea

$$\Gamma(\mathbf{C}) = \{B(x_1, \ldots, x_n) \mid \mathbf{C}(x_1, \ldots, x_n) = 1\}.$$

[5]Pentru detalii a se consulta *A. Atanasiu - Arhitectura Calculatoarelor*, editura InfoData Cluj, 2007.

Circuitul $\mathbf{C}$ fiind monoton, $\Gamma(\mathbf{C})$ este o mulţime monotonă de părţi ale lui $\mathcal{P}$.

Fiind dată o mulţime monotonă $\mathcal{A} \subseteq \mathcal{P}$, se poate construi uşor un circuit monoton $\mathbf{C}$ cu $\Gamma(\mathbf{C}) = \mathcal{A}$. Un exemplu de construcţie este următorul:

Fie $\mathcal{A}_{min}$ o bază a lui $\mathcal{A}$. Vom construi formula booleană (în forma normal disjunctivă)

$$\bigvee_{B \in \mathcal{A}_{min}} \left(\bigwedge_{P_i \in B} P_i \right)$$

Fiecare clauză din această formă normală este legată printr-o poartă AND, iar disjuncţia finală corespunde unei porţi OR.
Numărul total de porţi folosite este $1 + card(\mathcal{A}_{min})$.

Fie acum $\mathbf{C}$ un circuit monoton care recunoaşte $\mathcal{A}$. Vom construi un algoritm care permite arbitrului D să construiască un sistem perfect de partajare a secretului cu structura de acces $\mathcal{A}$, similar schemei de partajare (n, n) - majoritară definită în secţiunea 4.2. Mulţimea secretelor este deci $\mathcal{K} = Z_q$ (q număr prim).

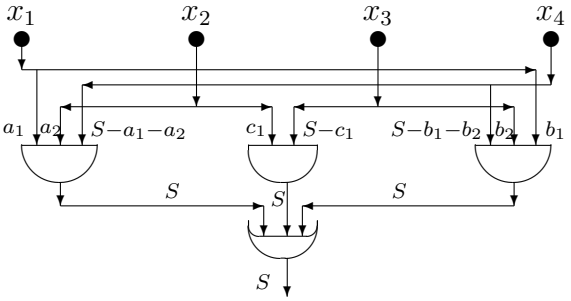
Algoritmul parcurge circuitul de la ieşire spre intrare, marcând recursiv cu $x_V \in \mathcal{K}$, fiecare arc V parcurs (în sens invers).
Iniţial, arcului care marchează ieşirea y i se atribuie valoarea $x_{out} = S$.
Formal, algoritmul este:

1. $x_{out} \leftarrow S$;
2. pentru orice poartă G din care iese un arc marcat x, iar arcele care intră sunt nemarcate, execută:
   (a) Dacă G este o poartă OR, atunci $x_V \leftarrow x$ pentru orice arc V care intră în G;
   (b) Dacă G este o poartă AND şi $V_1, \ldots, V_n$ sunt arcele care intră în G, atunci
      i. Alege aleator $x_{V,1}, \ldots, x_{V,n-1} \in Z_q$;
      ii. Calculează $x_{V,n} = x - \sum_{i=1}^{n-1} x_{V,i} \pmod q$;
      iii. Marchează arcul V_i cu $x_{V,i}$ $(1 \leq i \leq n)$.
      iv. D distribuie fiecărui participant P_i componenta S_i definită $S_i = \{x_{V,i} \mid V \text{ arc ce intră într-o poartă } AND\}$.

Exemplul 4.15. *Pentru mulţimea din Exemplul 4.1, avem* $\mathcal{A}_{min} = \{\{P_1, P_2, P_4\}, \{P_1, P_3, P_4\}, \{P_2, P_3\}\}$*, deci se poate asocia expresia booleană*

$$(P_1 \wedge P_2 \wedge P_4) \vee (P_1 \wedge P_3 \wedge P_4) \vee (P_2 \wedge P_3).$$

Circuitul monoton asociat este desenat mai jos; în paralel au fost marcate şi arcele, conform algoritmului descris:



Aici $a_1, a_2, b_1, b_2, c_1, c_2$ sunt numere alese aleator în Z_q. Fiecare participant primeşte drept componentă două numere: 1. *a_1 şi b_1 pentru P_1,*

2. *a_2 şi c_1 pentru P_2,*
3. *b_2 şi $S - c_1$ pentru P_3,*
4. *$S - a_1 - a_2$ şi $S - b_1 - b_2$ pentru P_4.*

Fiecare submulţime autorizată poate calcula valoarea lui S (modulo q).
Astfel, $\{P_1, P_2, P_4\}$ determină $S = a_1 + a_2 + (S - a_1 - a_2)$, submulţimea $\{P_1, P_3, P_4\}$ calculează $S = b_1 + b_2 + (S - b_1 - b_2)$, iar $\{P_2, P_3\}$ va calcula $S = c_1 + (S - c_1)$.

Să vedem acum ce se întâmplă cu mulţimile neautorizate.
Deoarece orice submulţime a unei mulţimi neautorizate sa va fi de asemenea neautorizată, este suficient să demonstrăm că mulţimile maximale neautorizate nu pot determina secretul, folosind informaţiile pe care le deţin.

Exemplul 4.16. *Revenind la exemplul anterior, mulţimile maximale neautorizate sunt $\{P_1, P_2\}, \{P_1, P_3\}, \{P_1, P_4\}, \{P_2, P_4\}, \{P_3, P_4\}$. În fiecare caz, pentru determinarea secretului S lipseşte o informaţie definită aleator.*

De exemplu, grupul $\{P_1, P_2\}$ deţine informaţiile a_1, a_2, b_1 şi c_1. Pentru a reconstitui S ar avea nevoie cel puţin de numărul $S - a_1 - a_2$, sau de $S - c_1$.

Sisteme cu aceeaşi structură de acces pot fi obţinute folosind şi alte circuite.

Exemplul 4.17. *Să reluăm Exemplul 4.1 şi să rescriem expresia booleană sub formă normal conjunctivă:*

$$(P_1 \vee P_2) \wedge (P_1 \vee P_3) \wedge (P_2 \vee P_3) \wedge (P_2 \vee P_4) \wedge (P_3 \vee P_4)$$

Construind schema de partajare corespunzătoare acestei expresii, vom avea următoarea distribuţie a componentelor (omitem detaliile):

1. *P_1 primeşte $S_1 = \{a_1,\ a_2\}$;*
2. *P_2 primeşte $S_2 = \{a_1,\ a_3,\ a_4\}$;*
3. *P_3 primeşte $S_3 = \{a_2,\ a_3,\ S - a_1 - a_2 - a_3 - a_4\}$;*
4. *P_4 primeşte $S_4 = \{a_4,\ S - a_1 - a_2 - a_3 - a_4\}$.*

Teorema 4.4. *Fie* $\mathbf{C}$ *un circuit combinaţional monoton. El defineşte o schemă perfectă de partajare a secretului, a cărui structură de acces este* $\mathcal{A} = \Gamma(\mathbf{C})$.

Demonstraţie. Vom folosi o inducţie după numărul de porţi din circuitul $\mathbf{C}$.

i. Cazul când $\mathbf{C}$ are o singură poartă este banal: dacă poarta este OR, fiecare participant deţine secretul S şi structura de acces este $\mathcal{A} = 2^{\mathcal{P}} \setminus \{\emptyset\}$; dacă poarta este AND şi are n intrări, se obţine sistemul (n, n)-majoritar definit anterior.

ii. Să presupunem că pentru $j > 1$, orice circuit $\mathbf{C}$ cu mai puţin de j porţi verifică teorema, şi fie $\mathbf{C}$ un circuit cu j porţi. Vom considera ultima poartă G a acestui circuit (din care iese rezultatul y). Ea nu poate fi decât OR sau AND. Dacă G este o poartă OR, să trasăm cele n arce $V_1, V_2, \ldots, V_n$ care intră în G. Acestea sunt arcele de ieşire din n circuite $\mathbf{C_i}$; conform ipotezei de inducţie, fiecare astfel de circuit defineşte un subsistem de partajare a secretului, cu structura de acces $\mathcal{A}_i = \Gamma(\mathbf{C_i})$. Vom avea – evident

$$\Gamma(\mathbf{C}) = \bigcup_{i=1}^{n} \mathcal{A}_i.$$

Cum valoarea S a secretului se atribuie fiecărui arc V_i, sistemul va avea structura de acces $\Gamma(\mathbf{C})$.

Procedeul este similar dacă G este o poartă AND. În acest caz,

$$\Gamma(\mathbf{C}) = \bigcap_{i=1}^{n} \mathcal{A}_i.$$

Deoarece secretul S este repartizat peste toate arcele V_i conform unui sistem (n, n) - majoritar, sistemul total va admite $\Gamma(\mathbf{C})$ drept structură de acces. □

Când un grup autorizată B doreşte aflarea secretului S, el trebuie să ştie circuitul utilizat de D pentru construirea schemei şi să deducă de aici ce componente sunt necesare pentru parcurgerea arcelor respective.
Această informaţie trebuie să fie publică. Numai valorile componentelor S_i trebuie să fie secrete.

4.5 Rata de informaţie

Fie $\mathcal{P}$ o mulţime de participanţi şi $\mathcal{S}$ spaţiul tuturor componentelor posibile ale cheii. O distribuţie de componente este o funcţie

$$f : \mathcal{P} \longrightarrow \mathcal{S}$$

Ea codifică matematic modalitatea de repartizare a informaţiilor între participanţi. $f(P_i)$ va fi componenta distribuită participantului P_i $(1 \leq i \leq n)$.

Pentru fiecare $S \in \mathcal{K}$, fie $\mathcal{F}_S$ mulţimea tuturor distribuţiilor posibile ale secretului K. În general, informaţia $\mathcal{F}_S$ este publică. Definim

$$\mathcal{F} = \bigcup_{S \in \mathcal{K}} \mathcal{F}_S.$$

$\mathcal{F}$ este ansamblul complet al tuturor distribuţiilor posibile de secrete. Rolul arbitrului va fi de a selecta aleator un element $f \in \mathcal{F}_S$ şi de a distribui componentele în conformitate cu această alegere.

Pentru o submulţime $B \subseteq \mathcal{P}$ (autorizată sau nu) de participanţi, se defineşte

$$S(B) = \{f^B \mid f \in \mathcal{F}\},$$

unde funcţia $f^B : B \longrightarrow \mathcal{S}$ este restricţia distribuţiei de componente f la submulţimea B; ea este deci definită prin $f^B(P_i) = f(P_i), \quad \forall P_i \in B$.

Deci $S(B)$ este mulţimea tuturor distribuţiilor posibile ale componentelor la participanţii din submulţimea B.

Să facem o evaluare a performanţelor sistemelor perfecte de partajare a secretelor construite până acum (pe baza structurilor de acces monotone).

În cazul unei scheme de partajare (n, k) - majoritare, circuitul boolean construit pe baza expresiei scrise în forma normal disjunctivă (a se vedea secţiunea 4.4) are $1 + C_n^k$ porţi. Fiecare participant primeşte o componentă formată din C_{n-1}^{k-1} numere din Z_p.
Această partajare este foarte slabă comparativ cu schema Shamir (n, k) - majoritară, care oferă acelaşi rezultat folosind componente formate dintr-un singur număr.

Pentru măsurarea performanţelor sistemelor perfecte de partajare a secretelor, vom folosi un instrument numit *rată de informaţie*.

Definiţia 4.10. *Considerăm un sistem perfect de partajare a secretelor cu structura de acces $\mathcal{A}$. Rata de informaţie a unui participant P_i este prin definiţie*

$$\rho_i = \frac{log_2(card(\mathcal{K}))}{log_2(card(S(P_i)))}.$$

unde $S(P_i) \subseteq \mathcal{S}$ este mulţimea componentelor posibile pe care le poate primi participantul P_i.

Rata de informaţie a sistemului este

$$\rho = min\{\rho_i \mid 1 \leq i \leq n\}$$

Exemplul 4.18. *Să comparăm cele două scheme folosite în exemplele din secţiunea anterioară. Schema din Exemplul 4.15 are rata de informaţie $\rho = \dfrac{log_2 q}{log_2 q^2} = \dfrac{1}{2}$.*

Pentru schema din Exemplul 4.17, avem $\rho = \dfrac{log_2 q}{log_2 q^3} = \dfrac{1}{3}$.

Deci prima schemă are o rată de informaţie mai bună.

În general, dacă se construieşte un sistem de partajare a secretelor plecând de la un circuit monoton $\mathbf{C}$, rata sa de informaţie se obţine folosind următoarea teoremă:

Teorema 4.5. *Fie* **C** *un circuit combinaţional monoton. Există atunci o schemă perfectă de partajare a secretelor, cu structura de acces* $\mathcal{A} = \Gamma(\mathbf{C})$*, care admite ca rată de informaţie*

$$\rho = \max_{1 \leq i \leq n} \left\{ \frac{1}{r_i} \right\}$$

unde r_i *este numărul arcelor de intrare din circuit (pentru valorile* x_i*).*

Evident, este preferabilă o rată de informaţie cât mai mare. Valoarea ei este însă limitată superior, conform teoremei următoare:

Teorema 4.6. *Pentru orice schemă perfectă de partajare a secretelor cu structura de acces* $\mathcal{A}$*, rata de informaţie verifică inegalitatea* $\rho \leq 1$*.*

Demonstraţie. Să considerăm un sistem perfect de partajare a secretelor având structura de acces $\mathcal{A}$. Fie $B \in \mathcal{A}_{min}$ şi $P_j \in B$ un participant arbitrar. Definim $B' = B \setminus \{P_j\}$. Fie $g \in S(B)$. Cum $B' \in \mathcal{NA}$, distribuţia componentelor $g_{B'}$ nu dă nici o informaţie despre secretul S.
Deci, pentru orice $S \in \mathcal{K}$ există o distribuţie a componentelor $g^S \in \mathcal{F}$ astfel ca $g^S_{B'} = g_{B'}$.
Cum $B \in \mathcal{A}$, vom avea $g^S(P_j) \neq g^{S'}(P_j)$ pentru $S \neq S'$.
Deci $card(S(P_j)) \geq card(\mathcal{K})$, adică $\rho \leq 1$. □

O schemă de partajare cu $\rho = 1$ va fi numită "ideală". Ca un exemplu, schema de partajare majoritară Shamir are $\rho = 1$, deci este o schemă ideală.
În schimb, rata de informaţie pentru o schemă de partajare (n, k) - majoritară bazată pe circuite monotone construite cu forma normal disjunctivă este $\frac{1}{C_{n-1}^{k-1}}$, extrem de ineficientă când $1 < k < n$.

4.6 Schema de partajare a lui Brickell

Sistemul construit în această secţiune este cunoscut sub numele de *construcţia vectorială a lui Brickell*.

Fie $\mathcal{A}$ o structură de acces, q un număr prim, iar $d \geq 2$ un număr întreg. Fie

$$\phi : \mathcal{P} \longrightarrow Z_q^d$$

o funcţie cu proprietatea[6]

$$(1, 0, \ldots, 0) \in \langle \phi(P_i) \mid P_i \in B \rangle \quad \Longleftrightarrow \quad B \in \mathcal{A}. \tag{1}$$

Altfel spus, vectorul $(1, 0, \ldots, 0)$ este o combinaţie liniară de vectori din mulţimea $\{\phi(P_i) \mid P_i \in B\}$ dacă şi numai dacă B este o mulţime autorizată.

[6]S-a notat cu $\langle X \rangle$ spaţiul generat de mulţimea de vectori X.

Plecând de la această funcţie, vom construi o schemă de partajare a secretelor cu $\mathcal{K}= S(P_i) = Z_q$ $(1 \leq i \leq n)$.
Pentru orice $\mathbf{a} = (a_1, \ldots, a_d) \in Z_p^d$. vom defini o funcţie de distribuţie a componentelor $f_{\mathbf{a}} : \mathcal{P} \longrightarrow \mathcal{S}$ prin

$$f_{\mathbf{a}}(x) = \mathbf{a} \cdot \phi(x)$$

Schema Brickell de partajare a secretelor este :

1. Pentru $1 \leq i \leq n$, D atribuie lui P_i vectorul $\phi(P_i) \in Z_q^d$.
   Aceşti vectori sunt publici.
2. Pentru partajarea secretului $S \in Z_q$, D alege $d-1$ elemente aleatoare $a_2, \ldots, a_d \in Z_q$.
3. Folosind vectorul $\mathbf{a} = (S, a_2, \ldots, a_d)$, arbitrul calculează componentele
   $$S_i = \mathbf{a} \cdot \phi(P_i) \quad (1 \leq i \leq n)$$
4. Pentru $i = 1, 2, \ldots, n$ arbitrul D trimite componenta S_i participantului P_i.

Vom avea rezultatul următor:

Teorema 4.7. *Dacă ϕ verifică proprietatea* (1)*, mulţimea distribuţiilor de componente $\mathcal{F}_S$, $S \in \mathcal{K}$ formează o schemă perfectă de partajare a secretelor, cu structura de acces $\mathcal{A}$.*

Demonstraţie. Să arătăm întâi că dacă B este o mulţime autorizată, participanţii lui B pot calcula secretul S. Deoarece $(1, 0, \ldots, 0) \in \langle \phi(P_i) \mid P_i \in B \rangle$, putem scrie

$$(1, 0, \ldots, 0) = \sum_{\{i | P_i \in B\}} c_i \phi(P_i)$$

unde $c_i \in Z_q$.
Fie S_i componenta lui P_i. Vom avea $S_i = \mathbf{a} \cdot \phi(P_i)$, unde $\mathbf{a}$ este vectorul necunoscut ales de D, iar $S = a_1 = \mathbf{a} \cdot (1, 0, \ldots, 0)$. Atunci

$$S = \sum_{\{i | P_i \in B\}} c_i \, \mathbf{a} \cdot \phi(P_i)$$

Membrii grupului B pot reconstitui deci secretul

$$S = \sum_{\{i | P_i \in B\}} c_i S_i$$

Ce se întâmplă dacă B nu este un grup autorizat ?
Fie e dimensiunea spaţiului vectorial $\langle \phi(P_i) \mid P_i \in B \rangle$ (evident, $e \leq card(B)$).
Fie $S \in \mathcal{K}$ şi să considerăm sistemul liniar

$$\begin{aligned} \phi(P_i) \cdot \mathbf{a} &= S_i \quad \forall P_i \in B \\ (1, 0, \ldots, 0) \cdot \mathbf{a} &= S \end{aligned}$$

cu necunoscutele $a_1, \ldots, a_d$.

Matricea sistemului are rangul $e + 1$ deoarece $(1, 0, \ldots, 0) \notin \langle \phi(P_i) \mid P_i \in B \rangle$. Deci, independent de valoarea lui S, spaţiul soluţiilor are gradul $d - e - 1$, adică există q^{d-e-1} distribuţii de componente în fiecare $\mathcal{F}_S$, consistente cu componentele participanţilor din B. □

Schema de partajare (n, k) - majoritară a lui Shamir este un caz particular al acestei construcţii.
Într-adevăr, fie $d = k$ şi $\phi(P_i) = (1, x_i, x_i^2, \ldots x_i^{k-1})$, pentru $1 \leq i \leq n$, unde x_i este coordonata x dată de P_i. Sistemul obţinut este echivalent cu sistemul din schema lui Shamir.

Un alt rezultat general se referă la structurile de acces care admit ca bază un ansamblu de perechi care definesc un graf multipartit complet.

Teorema 4.8. *Fie $G = (V, E)$ un graf multipartit complet. Atunci există un sistem perfect de partajare a secretelor, ideal, cu structura de acces $\overline{E}$ peste mulţimea V de participanţi.*

Demonstraţie. Fie $V_1, \ldots, V_s$ componentele lui G şi $x_1, \ldots, x_s \in Z_q$ distincte ($q \geq s$).
Să considerăm $d = 2$.
Pentru fiecare participant $v \in V_i$ se defineşte $\phi(v) = (x_i, 1)$. Proprietatea (1) se verifică imediat, deci – conform Teoremei 4.7 – afirmaţia este demonstrată. □

Vom aplica acest rezultat considerând toate structurile de acces posibile pentru $n = 4$ participanţi.

Va fi suficient să luăm în calcul numai structurile a căror bază nu se poate partiţiona în două mulţimi nevide. De exemplu, $\mathcal{A}_{min} = \{\{P_1, P_2\}, \{P_3, P_4\}\}$ poate fi partiţionată în $\{\{P_1, P_2\}\} \cup \{\{P_3, P_4\}\}$, fiecare cu dezvoltarea sa independentă, deci nu o vom lua în considerare.

O listă completă a structurilor de acces neizomorfe pentru 2, 3 sau 4 participanţi este dată în Tabelul 4.1 (s-a notat cu ρ^* valoarea maximă a ratei de informaţie pentru structura respectivă). Se pot construi scheme de partajare ideale pentru 10 din aceste 18 structuri de acces. Acestea sunt structuri majoritare sau structuri a căror bază este un graf multipartit, pentru care se aplică Teorema 4.8.

Exemplul 4.19. *Să considerăm structura de acces cu numărul 9 din Tabelul 4.1; deci $d = 2$ şi $q \geq 3$. Definim aplicaţia ϕ în felul următor*

$$\phi(P_1) = (0, 1), \quad \phi(P_2) = (0, 1), \quad \phi(P_3) = (1, 1), \quad \phi(P_4) = (1, 2)$$

Tabelul 4.1: Structuri de acces cu maxim 4 participanţi

Nr.crt	n	$\mathcal{A}_{min}$	ρ^*	Rezultate
1.	2	$\{P_1, P_2\}$	1	$(2,2)$- majoritar
2.	3	$\{P_1, P_2\}, \{P_2, P_3\}$	1	$\mathcal{A}_{min} \simeq K_{1,2}$
3.	3	$\{P_1, P_2\}, \{P_2, P_3\}, \{P_1, P_3\}$	1	$(3,2)$ - majoritar
4.	3	$\{P_1, P_2, P3\}$	1	$(3,3)$- majoritar
5.	4	$\{P_1, P_2\}, \{P_2, P_3\}, \{P_3, P_4\}$	2/3	
6.	4	$\{P_1, P_2\}, \{P_1, P_3\}, \{P_1, P_4\}$	1	$\mathcal{A}_{min} \simeq K_{1,3}$
7.	4	$\{P_1, P_2\}, \{P_1, P_4\}, \{P_2, P_3\}, \{P_3, P_4\}$	1	$\mathcal{A}_{min} \simeq K_{2,2}$
8.	4	$\{P_1, P_2\}, \{P_2, P_3\}, \{P_2, P_4\}, \{P_3, P_4\}$	2/3	
9.	4	$\{P_1, P_2\}, \{P_1, P_3\}, \{P_1, P_4\}, \{P_2, P_3\}, \{P_2, P_4\}$	1	$\mathcal{A}_{min} \simeq K_{1,1,2}$
10.	4	$\{P_1, P_2\}, \{P_1, P_3\}, \{P_1, P_4\}, \{P_2, P_3\}, \{P_2, P_4\}, \{P_3, P_4\}$	1	$(4,2)$ - majoritar
11.	4	$\{P_1, P_2, P_3\}, \{P_1, P_4\}$	1	
12.	4	$\{P_1, P_3, P_4\}, \{P_1, P_2\}, \{P_2, P_3\}$	2/3	
13.	4	$\{P_1, P_3, P_4\}, \{P_1, P_2\}, \{P_2, P_3\}, \{P_2, P_4\}$	2/3	
14.	4	$\{P_1, P_2, P_3\}, \{P_1, P_2, P_4\}$	1	
15.	4	$\{P_1, P_2, P_3\}, \{P_1, P_2, P_4\}, \{P_3, P_4\}$	1	
16.	4	$\{P_1, P_2, P_3\}, \{P_1, P_2, P_4\}, \{P_1, P_3, P_4\}$	1	
17.	4	$\{P_1, P_2, P_3\}, \{P_1, P_2, P_4\}, \{P_1, P_3, P_4\}, \{P_2, P_3, P_4\}$	1	$(4,3)$- majoritar
18.	4	$\{P_1, P_2, P_3, P_4\}$	1	$(4,4)$- majoritar

Aplicând Teorema 4.8 se obţine o structură perfectă de partajare a secretelor, ideală pentru acest tip de acces.

În Tabelul 4.1 rămân de analizat 8 structuri de acces. Pentru 4 din ele (structurile 11, 14, 15 şi 16) se poate utiliza schema lui Brickell.

Exemplul 4.20. *Pentru structura de acces* 11 *vom considera* $d = 3$ *şi* $q \geq 3$. *Definiţia lui* ϕ *este*

$$\phi(P_1) = (0,1,0), \quad \phi(P_2) = (1,0,1), \quad \phi(P_3) = (0,1,-1), \quad \phi(P_4) = (1,1,0)$$

Calculând, se obţine
$\phi(P_4) - \phi(P_1) = (1,1,0) - (0,1,0) = (1,0,0)$ *şi*
$\phi(P_2) + \phi(P_3) - \phi(P_1) = (1,0,1) + (0,1,-1) - (0,1,0) = (1,0,0)$.

Deci $(1,0,0) \in \langle \phi(P_1), \phi(P_2), \phi(P_3) \rangle$ *şi* $(1,0,0) \in \langle \phi(P_1), \phi(P_4) \rangle$.

Mai rămâne de arătat că $(1,0,0) \notin \langle \phi(P_i) \mid P_i \in B \rangle$ *pentru orice mulţime maximală neautorizată* B.

Există numai trei astfel de mulţimi: $\{P_1, P_2\}$, $\{P_1, P_3\}$, $\{P_2, P_3, P_4\}$. *Pentru fiecare caz se arată că sistemul liniar asociat nu are soluţie.*

De exemplu, să considerăm sistemul

$$(1,0,0) = a_2\phi(P_2) + a_3\phi(P_3) + a_4\phi(P_4)$$

cu $a_2, a_3, a_4 \in Z_p$. *El este echivalent cu sistemul*

$$\begin{aligned} a_2 + a_4 &= 1 \\ a_3 + a_4 &= 0 \\ a_2 - a_3 &= 0 \end{aligned}$$

care nu are soluţie.

Exemplul 4.21. *Pentru structura de acces* 14 *vom defini* $d = 3$, $q \geq 2$, *iar* ϕ *va fi definită:*

$$\phi(P_1) = (0,1,0), \quad \phi(P_2) = (1,0,1), \quad \phi(P_3) = (0,1,1), \quad \phi(P_4) = (0,1,1)$$

Proprietatea (1) *se verifică imediat; deci se poate aplica Teorema 4.8.*

În mod similar se pot construi sisteme perfecte de partajare a secretelor ideale pentru structurile 15 şi 16.

Cele patru sisteme rămase nu admit construcţia unor astfel de sisteme.

4.7 Construcţia prin descompunere

Prezentăm aici o altă modalitate de construire a schemelor de partajare a secretelor, remarcabilă prin performanţele rezultatelor, care maximizează rata de informaţie.

Definiţia 4.11. *Fie* $\mathcal{A}$ *o structură de acces cu baza* $\mathcal{A}_{min}$ *şi* $\mathcal{K}$ *o mulţime de secrete. O* $\mathcal{K}$ *- descompunere ideală a lui* $\mathcal{A}_{min}$ *este un set* $\{\Gamma_1, \ldots, \Gamma_w\}$ *cu proprietăţile*

1. $\Gamma_i \subseteq \mathcal{A}_{min} \quad (1 \leq i \leq w)$;
2. $\bigcup_{i=1}^{w} \Gamma_i = \mathcal{A}_{min}$;
3. $\forall\, i\ (1 \leq i \leq w)$ *există un sistem perfect de partajare a secretelor, ideal, cu mulţimea de secrete* $\mathcal{K}$, *peste mulţimea de participanţi* $\mathcal{P}_i = \bigcup_{B \in \Gamma_i} B$.

Pentru o $\mathcal{K}$ - descompunere ideală a structurii de acces $\mathcal{A}$ se poate construi uşor o schemă perfectă de partajare a secretelor.

Teorema 4.9. *Fie $\mathcal{A}$ o structură de acces cu baza $\mathcal{A}_{min}$, $\mathcal{K}$ o mulţime de secrete şi o $\mathcal{K}$ - descompunere ideală $\{\Gamma_1, \ldots, \Gamma_w\}$ a lui $\mathcal{A}$.*
Pentru fiecare participant P_i, fie $R_i = card\{s|\ P_i \in \mathcal{P}_s\}$.

Există atunci un sistem perfect de partajare a secretelor cu structură de acces $\mathcal{A}$ şi rată de informaţie $\rho = 1/R$, unde $R = \max\limits_{1 \le i \le w} \{R_i\}$.

Demonstraţie. Pentru $1 \le i \le w$ există un sistem ideal de structură de acces de bază Γ_i peste mulţimea secretelor $\mathcal{K}$. Notăm $\mathcal{F}_i$ mulţimea distribuţiilor componentelor sale.

Vom construi un sistem cu structură de acces $\mathcal{A}$ peste mulţimea $\mathcal{K}$.
Mulţimea distribuţiilor componentelor sale este generată după regula: dacă arbitrul D doreşte să împartă secretul S (în cazul $1 \le i \le w$), el va genera aleator o distribuţie de componente $f_i \in \mathcal{F}_S^i$ şi va distribui efectiv aceste componente participanţilor din $\mathcal{P}_i$.

Se verifică uşor că acest sistem este perfect. Să determinăm rata sa de informaţie. Vom avea $card(S(P_i)) = [card(\mathcal{K})]^{R_i}$ pentru orice i $(1 \le i \le n)$. Deci $\rho_i = 1/R_i$ şi

$$\rho = \frac{1}{max\{R_i \mid 1 \le i \le n\}},$$

ceea ce încheie demonstraţia. □

O generalizare a acestui rezultat – pentru s $\mathcal{K}$ - descompuneri ideale – se bazează pe teorema

Teorema 4.10. *(Construcţia prin descompunere): Fie $\mathcal{A}$ o structură de acces de bază $\mathcal{A}_{min}$, $s \ge 1$ un număr întreg, şi $\mathcal{K}$ un set de secrete. Presupunem că s-a construit o $\mathcal{K}$ - descompunere ideală $\mathcal{D}_j = \{\Gamma_{j,1}, \ldots, \Gamma_{j,w_j}\}$ a lui $\mathcal{A}_{min}$, şi fie $\mathcal{P}_{j,k}$ mulţimea participanţilor la structura de acces $\Gamma_{j,k}$. Pentru fiecare participant P_i definim*

$$R_i = \sum_{j=1}^{s} card\{k \mid P_i \in \mathcal{P}_{j,k}\}.$$

Există atunci o schemă perfectă de partajare a secretelor, cu structura de acces $\mathcal{A}$, a cărei rată de informaţie este $\rho = s/R$, unde $R = \max\limits_{1 \le i \le n}(R_i)$.

Demonstraţie. Pentru $1 \le j \le s$ şi $1 \le k \le w$ se poate construi o schemă de partajare ideală, cu baza $\Gamma_{j,k}$ şi mulţimea de secrete $\mathcal{K}$. Vom nota $\mathcal{F}_{j,k}$ mulţimea corespunzătoare de distribuţii a componentelor.

Vom construi un sistem cu structura de acces $\mathcal{A}$ şi mulţimea de secrete $\mathcal{K}^s$. Mulţimea sa de distribuţii de componente $\mathcal{F}$ se generează astfel: dacă arbitrul D doreşte să partiţioneze secretul $S = (S_1, \ldots, S_s)$ atunci – pentru fiecare k $(1 \le k \le w)$ – el va genera aleator o distribuţie de componente $f^{j,k} \in \mathcal{F}_{S_j}^{j,k}$, pe care le distribuie efectiv participanţilor din $\mathcal{P}_{j,k}$.

În continuare se repetă demonstraţia Teoremei 4.9. □

Exemplul 4.22. *Să considerăm structura de acces* 5 *din Tabelul 4.1, a cărei bază nu este un graf multipartit complet.*

Fie q un număr prim şi să considerăm două Z_q - descompuneri:

$\mathcal{D}_1 = \{\Gamma_{1,1}, \Gamma_{1,2}\}$ *cu* $\begin{array}{rcl} \Gamma_{1,1} & = & \{\{P_1, P_2\}\} \\ \Gamma_{1,2} & = & \{\{P_2, P_3\}, \{P_3, P_4\}\} \end{array}$

şi

$\mathcal{D}_2 = \{\Gamma_{2,1}, \Gamma_{2,2}\}$ *cu* $\begin{array}{rcl} \Gamma_{2,1} & = & \{\{P_1, P_2\}, \{P_2, P_3\}\} \\ \Gamma_{2,2} & = & \{\{P_3, P_4\}\} \end{array}$

Aceste descompuneri corespund lui K_2 şi $K_{1,2}$, deci sunt descompuneri ideale. Ambele oferă o rată de informaţie $\rho = 1/2$. Dacă le vom combina conform Teoremei 4.10 cu $s = 2$, vom obţine o rată de informaţie maximă $\rho = 2/3$.

Luând ca bază Teorema 4.8, putem obţine efectiv un astfel de sistem. D alege aleator patru elemente $b_{1,1}, b_{1,2}, b_{2,1}, b_{2,2} \in Z_q$. Pentru o cheie $S = (S_1, S_2) \in Z_q \times Z_q$, arbitrul va distribui componentele astfel:

1. *P_1 primeşte $\{b_{1,1},\ b_{2,1}\}$;*
2. *P_2 primeşte $\{b_{1,1} + S_1,\ b_{1,2},\ b_{2,1} + S_2\}$;*
3. *P_3 primeşte $\{b_{1,2} + S_1,\ b_{2,1},\ b_{2,2}\}$;*
4. *P_4 primeşte $\{b_{1,2},\ b_{2,2} + S_2\}$.*

(toate calculele sunt efectuate în Z_q).

Exemplul 4.23. *Fie structura de acces 8 din Tabelul 4.1. Vom considera $\mathcal{K} = Z_q$ pentru un număr prim $q \geq 3$. Vom utiliza două $\mathcal{K}$ - descompuneri ideale*

$\mathcal{D}_1 = \{\Gamma_{1,1}, \Gamma_{1,2}\}$ *cu* $\begin{array}{rcl} \Gamma_{1,1} & = & \{\{P_1, P_2\}\} \\ \Gamma_{1,2} & = & \{\{P_2, P_3\}, \{P_2, P_4\}, \{P_3, P_4\}\} \end{array}$

şi

$\mathcal{D}_2 = \{\Gamma_{2,1}, \Gamma_{2,2}\}$ *cu* $\begin{array}{rcl} \Gamma_{2,1} & = & \{\{P_1, P_2\}, \{P_2, P_3\}, \{P_2, P_4\}\} \\ \Gamma_{2,2} & = & \{\{P_3, P_4\}\} \end{array}$

$\mathcal{D}_1$ corespunde lui K_2 şi K_3, iar $\mathcal{D}_2$ – lui K_2 şi $K_{1,3}$; deci ambele sunt $\mathcal{K}$ - descompuneri. Aplicând Teorema 4.10 cu $s = 2$ se va obţine $\rho = 2/3$.

Similar exemplului precedent, o construcţie efectivă se realizează astfel:

D alege aleator patru elemente $b_{1,1}, b_{1,2}, b_{2,1}, b_{2,2} \in Z_q$.

Pentru o cheie $S = (S_1, S_2) \in Z_p^2$, el va distribui componentele astfel:

1. *P_1 primeşte $\{b_{1,1} + S_1,\ b_{2,1} + S_2\}$;*
2. *P_2 primeşte $\{b_{1,1},\ b_{1,2},\ b_{2,1}\}$;*
3. *P_3 primeşte $\{b_{1,2} + S_1,\ b_{2,1} + S_2,\ b_{2,2}\}$;*
4. *P_4 primeşte $\{b_{1,2} + 2S_1,\ b_{2,1} + S_2,\ b_{2,2} + S_2\}$.*

(toate calculele sunt efectuate în Z_q).

4.8 Scheme de partajare fără arbitru

Există posibilitatea ca un anumit secret să fie partiţionat fără a face apel la arbitru. Este o situaţie care apare frecvent în protocoale de partajare de chei. Evident, în acest

caz, secretul va fi considerat implicit şi va fi aflat doar atunci când este reconstituit de o mulţime autorizată de acces.

Ideea construirii unei astfel de partajări de secrete apare prima oară în lucrarea lui C. Meadows ([55]), dar o schemă funcţională este propusă de Ingermarsson şi Simmons în [42]. Aici, utilizatorul P_i alege un număr S_i care va fi una din cele n componente ale unui secret S, pe care îl partajează pentru ceilalţi utilizatori. Se obţine astfel o schemă de partajare $(n, n-1)$ - majoritară.

1. Fiecare participant P_i alege un număr aleator $S_i \in Z_q$ (q număr prim fixat şi public);
2. P_i generează aleator componentele $S_{i,j}$ astfel ca
$$S_i = \sum_{(j=1)\&(j\neq i)}^{n} S_{i,j} \quad (mod\ q)$$
3. P_i trimite fiecărui participant P_j $(1 \leq j \leq n,\ \ j \neq i)$ componenta $S_{i,j}$.

Deci, fiecare participant P_i va dispune de componenta

$$(S_i, S_{1,i}, \ldots, S_{i-1,i}, S_{i+1,i}, \ldots, S_{n,i})$$

Să presupunem că primii $n-1$ participanţi vor să recompună secretul. Ei vor calcula

$$S = \sum_{i=1}^{n-1} S_i + \sum_{i=1}^{n-1} S_{n,i} = \sum_{i=1}^{n-1} S_i + S_n = \sum_{i=1}^{n} S_i = S \quad (mod\ q)$$

Jackson, Martin şi O′ Keefe generalizează această schemă ([43]) la o mulţime de acces arbitrară $\mathcal{A}$.

1. Se foloseşte o schemă unanimă de ordin k pentru a construi componentele $S_1, \ldots, S_k$ ale unui secret S;
2. Utilizatorul P_i $(1 \leq i \leq k)$ împarte S_i (considerat ca un secret al cărui arbitru este) în componente, pentru o mulţime de acces $\mathcal{A}_i$, apoi împarte aceste componente utilizatorilor din $\mathcal{A}_i$.

Pentru construcţia mulţimilor de acces $\mathcal{A}_i$ se procedează în modul următor:

- Se ia mulţimea $\mathcal{A} \cup \{\{P_1\}\}$ şi se consideră baza ei; fie $\mathcal{A}_{1,min}$ această bază;
- $\mathcal{A}_1$ este mulţimea de acces generată de $\mathcal{A}_{1,min}$;

- Pentru $i = 2, \ldots, k$:
  - Se elimină utilizatorii $P_1, \ldots, P_{i-1}$ din $\mathcal{P}$; fie $\mathcal{A}'$ noua mulţime de acces;
  - Se construieşte $\mathcal{A}' \cup \{\{P_i\}\}$ şi se determină baza ei: $\mathcal{A}_{i,min}$;
  - $\mathcal{A}_i$ este mulţimea de acces generată de $\mathcal{A}_{i,min}$.

Exemplul 4.24. *Să construim o schemă de partajare* $(4, 3)$ *- majoritară, fără arbitru, pentru mulţimea de acces având baza*

$$\mathcal{A}_{min} = \{\{P_1, P_2, P_3\}, \{P_1, P_2, P_4\}, \{P_1, P_3, P_4\}, \{P_2, P_3, P_4\}\}$$

Cele trei baze sunt:

$$\mathcal{A}_{1,min} = \{\{P_1\}, \{P_2, P_3, P_4\}\}, \quad \mathcal{A}_{2,min} = \{\{P_2\}, \{P_3, P_4\}\}, \quad \mathcal{A}_{3,min} = \{\{P_3\}, \{P_4\}\}$$

Utilizatorul P_1*:*

1. *Generează aleator* S_1*;*
2. *Construieşte – cu un algoritm de partajare majoritară – componentele* $S_{1,2}, S_{1,3}, S_{1,4}$ *ale secretului* S_1*;*
3. *Distribuie* $S_{1,j}$ *participantului* P_j $(j = 2, 3, 4)$.

Utilizatorul P_2*:*

1. *Generează aleator* S_2*;*
2. *Construieşte – cu un algoritm de partajare majoritară – componentele* $S_{2,3}, S_{2,4}$ *ale secretului* S_2*;*
3. *Distribuie* $S_{2,j}$ *participantului* P_j $(j = 3, 4)$.

Utilizatorul P_3*:*

1. *Generează aleator* S_3*;*
2. *Trimite* $S_{3,4} = S_3$ *participantului* P_4.

Deci,secretul este $S = S_1 + S_2 + S_3$ *cu* $S_1 = S_{1,2} + S_{1,3} + S_{1,4}$, $S_2 = S_{2,3} + S_{2,4}$, $S_3 = S_{3,4}$ *şi*
P_1 *deţine* $\{S_1\}$,
P_2 *deţine* $\{S_2, S_{1,2}\}$
P_3 *deţine* $\{S_3, S_{1,3}, S_{2,3}\}$,
P_4 *deţine* $\{S_{1,4}, S_{2,4}, S_{3,4}\}$.

Dacă – de exemplu – mulţimea autorizată de acces $\{P_1, P_3, P_4\}$ *vrea să găsească secretul, va calcula*

$$S = S_1 + S_3 + S_{2,3} + S_{2,4}$$

4.9 Scheme de partajare verificabile

În toate schemele prezentate până acum s-a presupus că părţile implicate (arbitrul şi participanţii) se comportă onest. Cazul când arbitrul D trişează este studiat prima oară de Chor, Goldwasser, Micali şi Awerbuch ([20]); ei introduc şi noţiunea de *schemă de partajare verificabilă*, unde fiecare participant poate verifica dacă primit o componentă validă.

4.9.1 Schema de partajare a lui Feldman

P. Feldman propune în [31] o schemă de verificare a sistemului de partajare Shamir. Aceasta este:

1. Se generează numerele prime p, q astfel ca $q|(p-1)$; fie $\alpha \in Z_p^*$ un element de ordin q. Toate aceste valori sunt publice;
2. D generează polinomul
$$a(X) = a_0 + a_1 X + \ldots + a_{k-1} X^{k-1} \in Z_q[X]$$
cu $a_0 = S$; face publice elementele $\alpha_i = \alpha^{a_i} \ (mod\ p), \quad (0 \leq i \leq k-1)$;
3. D distribuie (prin canal securizat) către fiecare participant P_i componenta $S_i = a(i) \quad (i = 1, \ldots, n)$;

Fiecare participant verifică corectitudinea componentei primite S_i testând dacă are loc egalitatea

$$\alpha^{S_i} \ (mod\ p) \ = \ \prod_{j=0}^{k-1} \alpha_j^{i^j} \ (mod\ p)$$

Observaţia 4.5. *Schema lui Feldman poate fi construită pe un caz general, utilizând o funcţie homomorfă*[7] f. *Elementele* $f(a_0), f(a_1), \ldots, f(a_{k-1})$ *sunt publice, iar consistenţa componentei* S_i *se verifică pe baza egalităţii*

$$f(S_i) = f(a_0) \cdot f(a_1)^{i^1} \ldots f(a_{k-1})^{i^{k-1}}$$

Sistemul prezentat mai sus a folosit funcţia $f : Z_q \longrightarrow Z_p, \quad f(x) = \alpha^x \ (mod\ p)$, *cu* p, q *numere prime,* $q|(p-1)$ *şi* $\alpha \in Z_p^*$ *de ordin* q.

[7] O funcţie homomorfă are proprietatea $(\forall x, y) \ \ f(x+y) = f(x) \cdot f(y)$.

4.9.2 Schema lui Pedersen

Schema lui Feldman are dezavantajul că valoarea α^S este publică, şi deci confidenţialitatea secretului depinde direct de problema logaritmului discret. Pentru a elimina această (posibilă) falie, Pedersen propune următoarea variantă, relativă tot la sistemul de partajare Shamir:

1. Se generează numerele prime p, q cu $q|(p-1)$ şi elementele $g, h \in Z_p^*$ de ordin q. Toate aceste numere sunt publice;
2. Arbitrul D calculează $E_0 = g^S h^t \ (mod\ p)$ unde $t \in Z_q$ este ales arbitrar;
3. D generează polinoamele

$$a(X) = S + a_1 X + \ldots + a_{k-1} X^{k-1} \in Z_[X],$$
$$b(X) = t + b_1 X + \ldots + b_{k-1} X^{k-1} \in Z_q[X],$$

   calculează valorile $E_i = g^{a_i} h^{b_i} \ (mod\ p)$ şi face public vectorul $(E_0, E_1, \ldots, E_n)$;
4. D distribuie (prin canal securizat) către fiecare participant componenta

$$S_i = (a(i), b(i)), \quad i = 1, \ldots, n$$

Fiecare utilizator P_i poate testa corectitudinea componentei $S_i = (s_i, t_i)$ primite, verificând egalitatea

$$g^{s_i} h^{t_i} = \prod_{j=0}^{k-1} E_j^{ij} \ \ (mod\ p)$$

Schema Pedersen are proprietăţi de liniaritate: astfel, dacă secretele A_1, A_2 sunt definite prin componentele $(s_{i,j}, t_{i,j})$, $i = 1, \ldots, n$, $j = 1, 2$, atunci, pentru orice valori $a_1, a_2 \in Z_p$ secretul $a_1 A_1 + a_2 A_2$ este definit de componentele

$$(a_1 \cdot s_{i,1} + a_2 \cdot s_{i,2}), \quad i = 1, 2, \ldots, n$$

toate calculele fiind efectuate în Z_q.

Această facilitate permite unele aplicaţii interesante în domenii adiacente, cum sunt gestiunea cheilor sau protocoale de vot electronic.

4.9.3 Scheme de partajare verificabile public

Schemele verificabile prezentate până acum permiteau doar participanţilor să îşi verifice corectitudinea propriilor componente primite. Este însă posibil să se solicite ca această

verificare să fie publică, putând fi efectuată de orice persoană interesată.
Prezentăm o astfel de schemă ([76]), propusă de Stadler în 1996 sub o formă generală:

1. Componentele corespunzătoare unui secret S sunt criptate de arbitrul D folosind cheile publice ale participanţilor, Aceste elemente

$$ES_i = e_{K_i}(S_i), \quad (1 \leq i \leq n)$$

sunt publice.

2. Se foloseşte un algoritm public $PubVerify$ pentru a testa validitatea componentelor criptate. Astfel, dacă pentru o mulţime de acces $A \in \mathcal{A}_{min}$ avem $PubVerify(\{ES_i \mid P_i \in A\}) = 1$, atunci participanţii din A pot – după ce decriptează componentele – să recompună secretul S.

Sunt diverse protocoale propuse pentru construirea algoritmilor $PubVerify$; majoritatea lor sunt de tipul *(Provocare, Răspuns)* şi vor fi studiate în Capitolul 7.

4.10 Exerciţii

4.1. *Construiţi o* $(5, 3)$ *- schemă de partajare Shamir peste* Z_{29}*, folosind polinomul de interpolare* $a(X) = 11X^2 + X + 18$ *şi cheile secrete* $x_i = 3^i \ (mod\ 29), \ i = 1, \ldots, 5$.
Recompuneţi secretul pentru mulţimea $\{P_2, P_3, P_5\}$.

4.2. *Se dă secvenţa* $7, 9, 11, 12, 13, 15, 17, 19$.

1. *Pentru ce valori ale lui* k *ea este un şir* $(7, k)$ *- Mignotte ?*
2. *Pentru fiecare astfel de* k*, să se construiască o schemă Mignotte de partajare a secretului* $S = \lfloor(\beta - \alpha)/2\rfloor$.

4.3. *Folosind graful multipartit complet* $K_{2,3,3}$ *şi valorile* $q = 13, \ r = 5, \ x_1 = 1, \ x_2 = 2, \ x_3 = 3$*, să se construiască o schemă de partajare pentru secretul* $S = 8$.

4.4. *Construiţi o schemă perfectă de partajare pentru structura de acces:*

$$\mathcal{A} = \{\{P_1\}, \{P_2, P_3\}, \{P_2, P_4, P_5\}, \{P_3, P_4, P_5\}\}$$

4.5. *Să se construiască circuite monotone pentru structurile de acces din exerciţiul anterior.*

4.6. *Construiţi sisteme de partajare perfecte cu rata de informaţie* $\rho = 1/2$*, folosind metoda circuitelor monotone, pentru structurile de acces ale căror baze sunt;*

1. $\mathcal{A}_{min} = \{\{P_1, P_2\}, \{P_2, P_3\}, \{P_2, P_4\}, \{P_3, P_4\}\}$.
2. $\mathcal{A}_{min} = \{\{P_1, P_3, P_4\}, \{P_1, P_2\}, \{P_2, P_3\}, \{P_2, P_4\}\}$.
3. $\mathcal{A}_{min} = \{\{P_1, P_2\}, \{P_1, P_3\}, \{P_2, P_3, P_4\}, \{P_2, P_4, P_5\}, \{P_3, P_4, P_5\}\}$.

4.7. *Construiţi sisteme de partajare perfecte folosind schema Brickell, pentru structurile de acces ale căror baze sunt:*

1. $\mathcal{A}_{min} = \{\{P_1, P_2, P_3\}, \{P_1, P_2, P_4\}, \{P_3, P_4\}\}$.
2. $\mathcal{A}_{min} = \{\{P_1, P_2, P_3\}, \{P_1, P_2, P_4\}, \{P_1, P_3, P_4\}\}$.
3. $\mathcal{A}_{min} = \{\{P_1, P_2\}, \{P_1, P_3\}, \{P_1, P_3\}, \{P_1, P_4, P_5\}, \{P_2, P_4, P_5\}\}$.

4.8. *Construiţi sisteme perfecte de partajare a secretelor pentru structurile de acces* 15 *şi* 16 *din Tabelul 4.1.*

4.9. *Folosind metoda descompunerilor, construiţi sisteme perfecte de partajare cu rata de informaţie specificată, pentru structurile de acces având bazele:*

1. $\mathcal{A}_{min} = \{\{P_1, P_3, P_4\}, \{P_1, P_2\}, \{P_2, P_3\}\},\ \rho = 3/5$.
2. $\mathcal{A}_{min} = \{\{P_1, P_3, P_4\}, \{P_1, P_2\}, \{P_2, P_3\}, \{P_2, P_4\}\},\ \rho = 4/7$.

Capitolul 5

Protocoale de gestiune a cheilor

5.1 Proprietăţi generale

Am văzut că sistemele bazate pe chei publice nu necesită un canal sigur pentru transmiterea unei chei private. Aceste avantaj este redus însă de faptul că un canal cu cheie publică este mult mai lent decât unul cu cheie simetrică, el necesitând protocoale de transmitere, scheme de autentificare etc.

În acest capitol vom aborda această problemă, de stabilire a unor modalităţi de a schimba mesaje între mai mulţi utilizatori, folosind protocoale rapide, fiecare utilizator având propria sa cheie privată.

Protocoalele de gestiune a cheilor acoperă diverse variante. Astfel există

1. Protocoalele de **distribuire a cheilor**: o autoritate generează o cheie K pe care o transmite mai departe altor entităţi (utilizatori).

2. Protocoalele de **punere de acord** (key agreement): toţi participanţii stabilesc o cheie K folosind scheme de partajare a secretelor peste un canal public.

Protocoalele de punere de acord pot fi:

- **implicite**: *Alice* este asigurată că nimeni înafară de *Bob* nu poate determina cheia K. De remarcat că acest lucru nu înseamnă că *Alice* este sigură că *Bob* are efectiv cheia K. Un astfel de protocol se numeşte *punere de acord autentificată* (AK).

- **explicite**: *Alice* este sigură că *Bob* a calculat efectiv cheia K.
  De obicei, aceste protocoale se obţin prin completarea punerilor de acord implicite cu protocoale de confirmare a cheii (de către *Bob*). Astfel de protocoale sunt numite AKC (protocoale de *punere de acord explicită cu confirmare*).

Baza de comunicare este o reţea (nesigură) care leagă n utilizatori. În general se presupune că există o autoritate T numită *arbitru*, considerată apriori ca fiind onestă; ea

verifică identităţile utilizatorilor, alegerea şi transmiterea cheilor iniţiale, precum şi alte probleme curente.

Deoarece reţeaua nu este sigură, participanţii trebuie să prevadă unele atacuri din partea lui *Oscar*. Acesta poate fi pasiv (doar interceptează mesaje) sau activ.

Un adversar activ este capabil:

- Să modifice un mesaj în cursul transmisiei;
- Să transmită mesaje vechi;
- Să încerce să se prezinte drept unul din utilizatorii reţelei.

Obiectivul lui *Oscar* este:

- Să îi facă pe *Alice* şi *Bob* să accepte o cheie *invalidă*;
- Să îi facă pe *Alice* şi *Bob* să creadă că au schimbat chei între ei.

Scopul unui protocol de *distribuţie a cheilor* sau de *punere de acord* este ca în final cei doi participanţi să deţină aceeaşi cheie K, necunoscută de ceilalţi utilizatori (exceptând eventual T).

Alte proprietăţi de securitate care pot fi avute în vedere (considerând că *Alice* şi *Bob* sunt oneşti):

- **Securitatea cheii**: Fiecare execuţie a protocolului de generare a cheii trebuie să genereze o cheie secretă unică – numită *cheie de sesiune*. Este important ca aceste chei să ofere lui *Oscar* doar o cantitate limitată de informaţii şi să limiteze cât mai mult posibil – în caz de compromitere – expunerea altor chei.
  Protocolul trebuie să realizeze acest deziderat chiar dacă *Oscar* a acumulat informaţii deduse din cheile de sesiune generate anterior.
- **Secretul cheilor ulterioare**: Compromiterea cheii secrete a unuia sau mai multor utilizatori să nu afecteze secretul cheilor de sesiune anterioare stabilite de utilizatorii oneşti. Adesea se face distincţie între situaţia când numai cheia secretă a lui *Alice* este compromisă, şi cea când cheile lui *Alice* şi *Bob* sunt compromise simultan.
- **Impersonalizarea cheii compromise**: Dacă *Oscar* a descoperit cheia secretă a lui *Alice*, el îi poate lua locul în protocoalele de gestiune a cheii (o "impersonalizează" pe *Alice*). Acest lucru nu trebuie însă să-l impersonalizeze pe *Bob* faţă de *Alice*.
- **Anonimitatea componentelor cheii**: *Bob* nu poate fi obligat să împartă o cheie cu o entitate pe care nu o cunoaşte; de exemplu, *Bob* poate crede (eronat) că împarte o cheie cu *Charlie* în timp ce alt utilizator – *Alice* – crede (corect) că cheia este partajată cu *Bob*.

Alte performanţe care pot fi urmărite la protocoalele de gestiune a cheilor:

1. Număr minim de mesaje interschimbate;
2. Număr cât mai mic de biţi transmişi;
3. Număr cât mai mic de calcule;
4. Existenţa unei faze de precalcul.

Pentru uşurinţa formalizării, în acest capitol vom nota utilizatorii (*Alice, Bob, Charlie*, ...) cu $A, B, C, \ldots$, iar autoritatea (arbitrul, serverul) cu T.

5.2 Protocoale de distribuire a cheilor

Schema generală pentru această clasă de protocoale de gestiune a cheilor este: pentru orice pereche de utilizatori (A, B), arbitrul T generează aleator o cheie $K_{A,B} = K_{B,A}$ pe care o transmite lui A şi B printr-un canal securizat (neutilizat pentru comunicaţiile uzuale). Această cheie va fi folosită ulterior de cei doi utilizatori pentru a inter-schimba mesaje.

Deci, în această fază, T generează C_n^2 chei pe care le distribuie celor n utilizatori. Fiecare utilizator trebuie să păstreze cheia sa, precum şi cele $n-1$ chei de comunicare cu ceilalţi utilizatori.

Această etapă este sigură, dar ridică o serie de probleme cum ar fi:

- Existenţa de canale securizate între T şi fiecare din cei n participanţi. De exemplu, pentru o reţea cu 1000 utilizatori sunt necesare $C_{1000}^2 = 499.500$ canale sigure de comunicaţie. Mai mult, la venirea unui utilizator nou trebuie securizate alte 1000 canale.
- Obligaţia pentru fiecare participant să stocheze $n-1$ chei şi să participe la C_n^2 transmisii de chei solicitate de server.
  Chiar pentru o reţea mică, acest lucru devine destul de dificil.

Scopul protocoalelor prezentate este de a micşora (fără a periclita siguranţa comunicării) cantitatea de informaţie care trebuie transmisă sau stocată.

5.2.1 Protocolul Blom

Fie n ($n \geq 3$) utilizatori şi p ($p \geq n$) un număr prim. Fiecare cheie este un element din Z_p^*. Se alege un număr întreg $k \in Z_{n-1}^*$.
Valoarea lui k este numărul maxim de intruşi împotriva contra cărora poate fi asigurată protecţia.

În procedeul Blom – definit iniţial în 1984 pe baza codurilor separabile cu distanţă maximă (MDS) şi simplificat în 1993 – T transmite fiecărui utilizator, prin canale securizate, $k+1$ elemente din Z_p (în loc de $n-1$ chei în varianta generală).

Fiecare pereche de utilizatori (A, B) poate calcula o cheie de sesiune $K_{A,B} = K_{B,A}$.

Condiţia de securitate este: orice coaliţie de k utilizatori – diferiţi de A şi B – nu poate obţine informaţii despre $K_{A,B}$.

Pentru cazul $k = 1$ protocolul Blom este:

1. T face publice: un număr prim p şi – pentru fiecare utilizator A – un număr $r_A \in Z_p$. Toate numerele r_A sunt distincte.
2. T generează aleator trei numere $a, b, c \in Z_p$ şi formează polinomul
$$f(x, y) = a + b(x + y) + cxy \pmod p$$
3. Pentru fiecare utilizator A, T determină polinomul
$$g_A(x) = f(x, r_A) \pmod p$$
şi transmite $g_A(x)$ lui U printr-un canal securizat.
4. Dacă A şi B doresc să comunice între ei, cheia lor secretă este
$$K_{A,B} = K_{B,A} = f(r_A, r_B)$$

Observaţia 5.1.

- *Aplicaţia $g_A(x)$ de la pasul 3 este o transformare afină, de forma*
$$g_A(x) = a_A + b_A x$$
unde
$$a_A = a + br_A \pmod p, \qquad b_A = b + cr_A \pmod p.$$
- *La pasul 4, utilizatorul A poate calcula cheia secretă $K_{A,B} = f(r_A, r_B) = g_A(r_B)$, iar B – în mod similar – $K_{B,A} = f(r_A, r_B) = g_B(r_A)$.*

Exemplul 5.1. *Să presupunem că $p = 17$ şi sunt 3 utilizatori A, B, C – având cheile publice $r_A = 12$, $r_B = 7$ şi respectiv $r_C = 1$.*
Presupunem că arbitrul alege $a = 8$, $b = 7$, $c = 2$. Atunci
$$f(x, y) = 8 + 7(x + y) + 2xy$$

Polinoamele g sunt

$$g_A(x) = 7 + 14x, \qquad g_B(x) = 6 + 4x, \qquad g_C(x) = 15 + 9x.$$

Cele trei chei private sunt

$$K_{A,B} = 3, \quad K_{A,C} = 4, \quad K_{B,C} = 10.$$

A poate calcula $K_{A,B}$ prin

$$g_A(r_B) = 7 + 14 \cdot 7 = 3 \pmod{17};$$

B poate calcula $K_{B,A}$ prin

$$g_B(r_A) = 6 + 4 \cdot 12 = 3 \pmod{17}.$$

Să arătăm că nici un utilizator nu poate obţine singur informaţii asupra cheilor private ale celorlaltor participanţi.

Teorema 5.1. *Protocolul Blom pentru $k = 1$ este necondiţionat sigur*[1] *contra oricărui atac individual.*

Demonstraţie. Să presupunem că utilizatorul C doreşte să calculeze cheia

$$K_{A,B} = a + b(r_A + r_B) + cr_Ar_B \pmod{p}$$

Valorile r_A şi r_B sunt publice, dar a, b, c sunt secrete. C cunoaşte propriile sale valori

$$a_C = a + br_C \pmod{p} \qquad b_C = b + cr_C \pmod{p}$$

care sunt coeficienţii propriului său polinom $g_C(x)$.

Vom arăta că informaţia deţinută de C este consistentă cu orice valoare posibilă $m \in Z_p$ a cheii $K_{A,B}$. Deci C nu poate obţine nici o informaţie asupra cheii $K_{A,B}$.
Să considerăm ecuaţia matricială în Z_p:

$$\begin{pmatrix} 1 & r_A + r_B & r_Ar_B \\ 1 & r_C & 0 \\ 0 & 1 & r_C \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} m \\ a_C \\ b_C \end{pmatrix}$$

Prima ecuaţie este de fapt $K_{A,B} = m$, iar celelalte două dau informaţia cunoscută de C despre a, b, c.

Determinantul matricii este

$$r_C^2 + r_Ar_B - (r_A + r_B)r_C = (r_C - r_A)(r_C - r_B) \pmod{p}$$

Cum r_C este distinct de r_A şi r_B, acest determinant este nenul, deci sistemul admite o soluţie unică (a, b, c).
Cu alte cuvinte, orice valoare $m \in Z_p$ este compatibilă cu informaţia deja deţinută de C. □

[1] Un protocol este *necondiţionat sigur* dacă securitatea sa nu depinde de ordinul de complexitate al calculelor necesare pentru spargerea sa.

În schimb, orice coaliţie între doi participanţi C, D poate conduce la aflarea cheii $K_{A,B}$. Aceştia dispun de informaţiile:

$$\begin{cases} a_C = a + br_C, & b_C = b + cr_C, \\ a_D = a + br_D, & b_D = b + cr_D \end{cases}$$

Avem un sistem de 4 ecuaţii cu 3 necunoscute, din care se poate afla imediat soluţia unică (a, b, c). După ce s-au aflat aceste trei valori, se poate construi polinomul $f(x, y)$, din care se poate determina mai departe restul informaţiei.

Protocolul Blom poate fi generalizat pentru a rezista la atacul unei alianţe de k utilizatori. Singura modificare se face la pasul 2, unde arbitrul foloseşte polinomul

$$f(x, y) = \sum_{i=0}^{k} \sum_{j=0}^{k} a_{i,j} x^i y^j \quad (mod\ p)$$

unde $a_{i,j} \in Z_p$ $(0 \leq i \leq k,\ 0 \leq j \leq k)$ şi $a_{i,j} = a_{j,i}$ pentru orice i şi j.

5.3 Protocolul Needham - Schroeder

Un protocol de distribuţie a cheilor, celebru prin implicaţiile sale în stabilirea de chei de comunicaţie este definit în 1978 de Roger Needham şi Michael Schroder ([62]).

Protocolul Needham - Schroeder (NS) a fost propus în două variante:

- Protocolul NS cu chei simetrice. El a fost destinat stabilirii unei chei de sesiune între doi utilizatori conectaţi pe o reţea şi a stat ulterior la baza construirii protocolului Kerberos.
- Protocolul NS cu chei publice. Intenţa sa a fost de a oferi o autentificare reciprocă a celor doi utilizatori care comunică prin reţea

5.3.1 Protocolul NS cu chei simetrice

În acest protocol, A iniţiază un protocol de comunicaţie cu B, iar T este un server – considerat de ambii utilizatori ca fiind "de încredere". Se mai folosesc:

- $K_{A,T}$ – o cheie simetrică, cunoscută de A şi T;
- $K_{B,T}$ – o cheie simetrică, cunoscută de B şi T;
- N_A, N_B – numere (nonces) generate aleator de A respectiv B.
- $K_{A,B}$ – cheia (simetrică) de sesiune între A şi B.

Observaţia 5.2. *În protocoalele de securitate, termenul "nonce" reprezintă "un număr folosit o singură dată". În mod uzual, el este un număr generat aleator sau pseudo-aleator prin algoritmi care asigură un grad sporit de securitate. Ideea este ca un astfel de număr să nu mai poată fi generat a doua oară sub nici o formă. Scopul este evitarea reuşirii unor atacuri de tip reluare sau dicţionar.*
Este utilizat frecvent în protocoalele de autentificare, de partajare a secretelor, de stabilirea de chei de sesiune sau de construcţie de amprente.

În acest protocol vom nota cu $\{\alpha\}_K$ criptarea mesajului α folosind cheia K.
Varianta de bază a protocolului NS este:

1. A trimite lui T tripletul (A, B, N_A).
2. T trimite lui A mesajul: $\{N_A, K_{A,B}, B, \{K_{A,B}, A\}_{K_{B,T}}\}_{K_{A,T}}$.
3. A decriptează mesajul şi transmite mai departe lui B: $\{K_{A,B}, A\}_{K_{B,T}}$.
4. B decriptează şi trimite spre A mesajul $\{N_B\}_{K_{A,B}}$.
5. A răspunde lui B cu $\{N_B - 1\}_{K_{A,B}}$.

Comentarii asupra paşilor efectuaţi de protocolul NS:

1. A anunţă serverul că doreşte stabilirea unui canal de comunicaţie cu B.
2. Serverul generează cheia $K_{A,B}$ pe care o retrimite lui A în două copii: una pentru uzul lui A, iar alta – criptată cu $K_{B,T}$ – pe care A o va forwarda lui B.

   Deoarece A poate solicita chei pentru comunicarea cu mai mulţi utilizatori, nonce-ul asigură faptul că serverul a răspuns la acest mesaj; de asemenea, includerea lui B îi spune lui A cu cine va partaja această cheie de sesiune.
3. A forwardează cheia lui B; acesta o poate decripta folosind cheia de comunicare cu serverul; în acest fel se asigură şi o autentificare a datelor.
4. Prin acest mesaj, B îi arată lui A că deţine cheia de sesiune $K_{A,B}$.
5. A arată prin acest mesaj că este conectat în continuare şi că deţine cheia de sesiune.

Protocolul este vulnerabil la un atac prin reluare (identificat de Denning şi Sacco). Dacă *Oscar* deţine o valoare compromisă (mai veche) pentru $K_{A,B}$, el poate returna lui B mesajul $\{K_{A,B}, A\}_{K_{B,T}}$. Acesta îl va accepta, neputând să distingă dacă ce a primit este o cheie nouă sau nu.
Această slăbiciune va fi corectată de protocolul Kerberos (secţiunea următoare) prin introducerea unei ştampile de timp sau de nonce-uri.

5.3.2 Protocolul NS cu chei publice

A şi B folosesc un server T care distribuie – la cerere – chei publice. Aceste chei sunt:

- (K_{PA}, K_{SA}) – componenta publică şi secretă a cheii lui A.
- (K_{PB}, K_{SB}) – componenta publică şi secretă a cheii lui B.
- (K_{PT}, K_{ST}) – componenta publică şi secretă a cheii serverului T, cu menţiunea că K_{ST} este folosită pentru criptare, iar K_{PT} – pentru decriptare (deci T foloseşte cheia sa pentru autentificare).

Protocolul este următorul:

1. A trimite lui T perechea (A, B) prin care solicită cheia publică a lui B.
2. T trimite lui A elementul $\{K_{PB}, B\}_{K_{ST}}$.
3. A generează nonce-ul N_A şi-l trimite lui B: $\{N_A, A\}_{K_{PB}}$.
4. B trimite lui T perechea (B, A) solicitând cheia publică a lui A.
5. T răspunde cu $\{K_{PA}, A\}_{K_{ST}}$.
6. B generează nonce-ul N_B şi-l trimite lui A: $\{N_A, N_B\}_{K_{PA}}$.
   Trimiterea lui N_A arată lui A că B deţine de cheia de decriptare K_{SB}.
7. A confirmă că deţine cheia de decriptare, trimiţând lui B elementul $\{N_B\}_{K_{PB}}$.

La sfârşitul protocolului, A şi B cunosc fiecare identităţile celuilalt şi nonce-urile N_A, N_B.

Securitatea protocolului

Varianta cu cheie publică a protocolului Needham - Schroeder nu rezistă atacului *man-in-the-middle*. Dacă *Oscar* (notat cu O) îl convinge pe A să iniţieze o sesiune cu el, poate retrimite mesajele lui B şi să-l convingă că este în contact cu A.
Ignorând traficul de informaţii (din protocol) cu serverul T, atacul se desfă-şoară astfel:

1. A trimite lui O mesajul $\{N_A, A\}_{K_{PO}}$.
2. O decriptează cu K_{SO}, după care trimite lui B mesajul $\{N_A, A\}_{K_{PB}}$.

3. B răspunde lui O (crezând că este A) cu mesajul $\{N_A, N_B\}_{K_{PA}}$.

4. O trimite acest mesaj mai departe, spre A.

5. A răspunde lui O cu $\{N_B\}_{K_{PO}}$.

6. O extrage de aici N_B şi-l trimite lui B, criptat cu cheia publică a acestuia: $\{N_B\}_{K_{PB}}$.

În final, B este convins că comunică cu A şi că N_A, N_B sunt cunoscute doar de el şi de A.

Atacul a fost prezentat prima oară în 1995 de Gavin Lowe. Acesta construieşte o variantă a protocolului, numită ulterior protocolul *Needham - Schroeder - Lowe*. Modificarea se referă la pasul 6 din protocol, unde mesajul $\{N_A, N_B\}_{K_{PA}}$ – trimis de B către A – este înlocuit cu $\{N_A, N_B, B\}_{K_{PA}}$.

5.4 Protocolul Kerberos

Kerberos este unul din cele mai răspândite sisteme de gestiune a cheilor – în special versiunile 4 şi 5 (ultimul, adoptat ca standard Internet *RFC* 1510). Protocolul se bazează parţial pe protocolul Needham - Schroeder, aducând unele îmbunătăţiri legate de securitate.

În acest protocol, fiecare utilizator A împarte cu T o cheie DES[2] notată K_A.
De asemenea, lui A i se asociază (în general sub controlul lui T) o secvenţă de informaţii care-l identifică în mod unic (nume, adresă, CNP, număr telefon etc). Această secvenţă se notează cu $ID(A)$.

La solicitarea lui A de a comunica cu B, arbitrul T efectuează următoarele operaţii:

- Generează o cheie K;
- Înregistrează ora H a cererii;
- Stabileşte o durată L de validitate a lui K; deci cheia de sesiune este validă în intervalul de timp $[T, T + L]$.

Tichetul cu informaţiile (K, H, L) sunt transmise de T lui A, apoi lui B. Protocolul Kerberos este prezentat pe pagina următoare.

În acest protocol, fiecare din cele patru mesaje m_i transmise are rolul său bine determinat.
Astfel, m_1 şi m_2 servesc la transmiterea confidenţială a cheii K. La rândul lor, m_3 şi m_4 asigură o confirmare a cheii; după primirea ei, A şi B sunt siguri că dispun de aceeaşi cheie de sesiune K.

Rolul lui H şi L este acela de protejare contra unui atac activ constând din înregistrarea unor mesaje vechi şi – ulterior – retransmiterea lor.

[2]Ultimele versiuni folosesc modul CBC de implementare a sistemului DES.

1. T generează K, H şi L.
2. T calculează mesajele
$$m_1 = \{K, ID(B), H, L\}_{K_A}, \quad m_2 = \{K, ID(A), H, L\}_{K_B}$$
pe care le trimite lui A.
3. A decriptează m_1 şi află $K, H, L, ID(B)$. Pe urmă calculează
$$m_3 = \{ID(A), H\}_K$$
şi trimite lui B mesajele m_2 şi m_3.
4. B află $K, H, L, ID(A)$ din decriptarea lui m_2. Cu ajutorul lui K decriptează m_3 şi află $H, ID(A)$.
Verifică dacă cele două valori pentru H şi $ID(A)$ sunt identice.
5. B calculează
$$m_4 = \{H + 1\}_K$$
pe care îl trimite lui A.
6. A decriptează m_4 şi verifică dacă mesajul obţinut este $H + 1$.

Există diverse opţiuni de implementare a tichetului (K, H, L), cum ar fi de exemplu:

- *Tichet eliberat pe termen lung.* Există riscul de a permite atacuri asupra cheii.
- *Tichet postdatat.* Riscul este ca *Oscar* să afle tichetul înainte de utilizarea sa; de aceea multe servere au permisiunea de a le putea respinge.
- *Tichet proxy.* Un client dă permisiune serverului de a iniţia anumite comunicaţii în numele său.

Una din slăbiciunile sistemului *Kerberos* constă în imposibilitatea unei bune sincronizări a ceasurilor utilizatorilor. În practică se admit anumite decalaje, stabilite de comun acord.

În plus, spargerea sistemului DES a condus – în unele standarde – la înlocuirea sistemului *Kerberos*.

5.5 Schimbul de chei Diffie - Hellman

Dacă nu se acceptă un furnizor universal de chei, atunci va trebui stabilit un protocol de punere de acord.
Primul şi cel mai cunoscut astfel de protocol este *protocolul de schimb de chei Diffie - Hellman.*
Definit în 1976 ([27]) şi folosit în diverse variante, el se bazează pe problema logaritmului discret.

5.5.1 Variante ale protocolului Diffie - Hellman

Pentru iniţializare, să considerăm:

- p – număr prim (de 1024 biţi – în protocoalele standard);
- q – divizor prim al lui $p-1$ (de 160 biţi);
- $\alpha \in Z_p^*$, element de ordin q.
- h – funcţie de dispersie criptografică (de exemplu $SHA1$);
- (sig_T, ver_T) – algoritm de semnătură a arbitrului T.

Utilizatorul A dispune de cheia secretă $a \in Z_q^*$ şi cheia publică $Y_A = \alpha^a$.

Pe baza componentelor publice (stocate în $ID(A)$) şi la solicitarea lui A, arbitrul T eliberează un certificat

$$Cert(A) = (ID(A), Y_A, sig_T(ID(A), Y_A))$$

care este de asemenea public.
Orice utilizator B poate verifica autenticitatea certificatului $Cert(A)$ şi poate obţine de aici – printre altele – cheia publică Y_A.

Aceleaşi elemente (b, $Y_B = \alpha^b$, $ID(B)$, $Cert(B)$) sunt generate pentru utilizatorul B.

Observaţia 5.3.

1. *Arbitrul nu trebuie să cunoască cheia secretă "a" pentru a produce certificatul. Când A intră în reţea, se generează un astfel de certificat, care poate fi păstrat în baza de date sau poate fi comunicat chiar de A la fiecare utilizare. Semnătura lui T permite oricui să verifice autenticitatea informaţiei pe care o conţine.*
2. *Înainte de a elibera $Cert(A)$, T se convinge că A posedă cheia secretă "a" corespunzătoare cheii publice Y_A (un astfel de protocol, numit "provocare - răspuns", este prezentat în Capitolul 7).*
   În acest mod se evită un atac prin care Oscar înregistrează Y_A drept cheia sa publică, îngreunând astfel posibilitatea ca un alt utilizator B să intre în contact cu A.

3. De asemenea, T face o "validare" a cheii publice Y_A, verificând relaţiile

$$1 < Y_A < p, \quad Y_A^q \equiv 1 \ (mod\ p)$$

Sunt două versiuni ale protocolului de bază Diffie - Hellman, în funcţie de durata valabilităţii cheii de sesiune generate:

Protocolul Diffie - Hellman de perioadă scurtă:

1. A generează aleator $x \in Z_q^*$ şi trimite lui B valoarea $R_A = \alpha^x \ \ (mod\ p)$;
2. B generează aleator $y \in Z_q^*$ şi trimite lui A valoarea $R_B = \alpha^y \ \ (mod\ p)$;
3. A calculează $K_{A,B} = R_B^x = \alpha^{xy}$;
4. B calculează $K_{B,A} = R_A^y = \alpha^{xy}$.

Acest protocol[3] asigură o autentificare implicită a cheii de sesiune. Avantajul său este acela că fiecare execuţie a protocolului generează o cheie de sesiune nouă.
Dezavantajul este că nu rezistă la un atac *man-in-the-middle*, neexistând nici o autentificare a utilizatorilor implicaţi.

Protocolul Diffie - Hellman de perioadă lungă:

1. A trimite lui B certificatul $Cert(A)$;
2. B trimite lui A certificatul $Cert(B)$;
3. A extrage Y_B din $Cert(B)$ şi calculează $K_{A,B} = Y_B^a = \alpha^{ab}$;
4. B extrage Y_A din $Cert(A)$ şi calculează $K_{B,A} = Y_A^b = \alpha^{ab}$.

Avantajul acestei scheme constă în autentificarea reciprocă a partenerilor. Dezavantajul este că la fiecare execuţie a protocolului se obţine aceeaşi cheie de sesiune; cheile se schimbă doar odată cu schimbarea certificatelor – după o perioadă de timp relativ lungă.

Exemplul 5.2. *Să presupunem $p = 25307$ şi $\alpha = 2$. Dacă luăm $a = 3578$, vom avea $Y_A = 2^{3578} = 6113 \ \ (mod\ 25307)$, valoare pusă în certificatul lui A.*

Să presupunem că B alege $b = 19956$; atunci $Y_B = 2^{19956} = 7984 \ \ (mod\ 25307)$.

A poate calcula cheia comună

$$K_{A,B} = 7984^{3578} = 3694 \quad (mod\ 25307)$$

[3]Schimbul de chei din sistemul *SSH* (Secure SHell) – care asigură comunicarea pentru sistemele bazate pe UNIX – are implementat protocolul Diffie - Hellman de perioadă scurtă.

Aceiaşi cheie este calculată şi de B:

$$K_{U,V} = 6113^{19956} = 3694 \quad (mod\ 25307)$$

Matsumoto, Takashima şi Imai ([54]) combină cele două protocoale într-unul singur:

Protocolul $MTI/C0$:

1. A generează aleator $x \in Z_q^*$ şi trimite lui B valoarea $T_A = Y_B^x \ \ (mod\ p)$;
2. B generează aleator $y \in Z_q^*$ şi trimite lui A valoarea $T_B = Y_A^y \ \ (mod\ p)$;
3. A calculează $K_{A,B} = (T_B)^{a^{-1}x} = \alpha^{xy}$;
4. B calculează $K_{B,A} = (T_A)^{a^{-1}y} = \alpha^{xy}$.

Este interesant că acest protocol – spre deosebire de variantele din care provine – nu asigură o autentificare implicită a cheii. Astfel, *Oscar* poate – printr-un atac de tip *man-in-the-middle* – să înlocuiască T_A şi T_B cu numărul 1.
Atunci A şi B vor obţine cheia 1, cheie cunoscută şi de *Oscar*.

5.5.2 Securitatea protocolului Diffie - Hellman

Să studiem securitatea acestui protocol contra unui atac (activ sau pasiv).

Semnătura lui T pe certificate împiedică producerea de informaţii publice false. Deci vor rămâne în discuţie numai atacurile pasive, care se reduc la problema:

"*C poate determina $K_{A,B}$ dacă nu este A sau B ?*"

Altfel spus: "Fiind date α^a şi α^b, ambele $(mod\ p)$, se poate calcula α^{ab} $(mod\ p)$?"

Aceasta este cunoscută şi sub numele de *problema Diffie - Hellman* (variante ale acestei probleme sunt prezentate în Anexa 2):

Fie $I = (p, \alpha, \beta, \gamma)$ unde p este număr prim, $\alpha \in Z_p$ este primitiv, iar $\beta, \gamma \in Z_p^*$.
Se poate determina $\beta^{log_\alpha \gamma}$ $(mod\ p)$? (sau – echivalent, $\gamma^{log_\alpha \beta}$ $(mod\ p)$)

Evident, securitatea protocolului Diffie - Hellman (de gestiune a cheilor) faţă de atacurile pasive este echivalentă cu dificultatea problemei Diffie - Hellman.

Dacă C poate calcula a (sau b) plecând de la Y_A (respectiv Y_B), el poate deduce $K_{A,B}$ aşa cum face A (respectiv B).
Aceasta conduce la rezolvarea unei probleme de logaritm discret. Deci, dacă problema logaritmului discret în Z_p este dificilă, atunci protocolul de punere de acord a cheii Diffie - Hellman nu poate fi atacat.

Conjectura este aceea că problema Diffie - Hellman este echivalentă cu problema logaritmului discret (aşa cum s-a conjecturat că spargerea sistemului RSA este echivalentă cu factorizarea unui număr).

Teorema 5.2. *A sparge sistemul de criptare El Gamal este echivalent cu a rezolva problema Diffie - Hellman.*

Demonstraţie. Să reamintim sistemul de criptare *El Gamal*:
O cheie este $K = (p, \alpha, a, \beta)$ unde $\beta = \alpha^a \ (mod\ p)$; a este secret, iar p, α, β sunt publice.
Criptarea unui mesaj $x \in Z_p$ se face alegând aleator un număr $k \in Z_{p-1}^*$; apoi

$$e_K(x, p) = (y_1, y_2)$$

unde

$$y_1 = \alpha^k \pmod p, \qquad y_2 = x\beta^k \pmod p.$$

Pentru $y_1, y_2 \in Z_p^*$, decriptarea este definită prin

$$d_K(y_1, y_2) = y_2(y_1^a)^{-1} \pmod p.$$

Să presupunem că dispunem de un algoritm $\mathcal{A}$ care rezolvă problema Diffie - Hellman şi încercăm un atac asupra mesajului criptat (y_1, y_2).
Aplicând $\mathcal{A}$ asupra intrărilor p, α, y_1 şi β avem:

$$\mathcal{A}(p, \alpha, y_1, \beta) = \mathcal{A}(p, \alpha, \alpha^k, \alpha^a) = \alpha^{ka} = \beta^k \pmod p$$

De aici rezultă $x = y_2(\beta^k)^{-1} \ (mod\ p)$, deci se poate decripta mesajul (y_1, y_2).

Invers, să presupunem că dispunem de un algoritm $\mathcal{B}$ de decriptare pentru sistemul *El Gamal*. Deci $\mathcal{B}$ admite la intrare $(p, \alpha, \beta, y_1, y_2)$ şi calculează

$$x = y_2 \left(y_1^{log_\alpha \beta}\right)^{-1} \pmod p$$

Fiind dată o apariţie $(p, \alpha, \beta, \gamma)$ a problemei Diffie - Hellman, se poate calcula uşor

$$\mathcal{B}(p, \alpha, \beta, \gamma, 1)^{-1} = 1 \left(\left(\gamma^{log_\alpha \beta}\right)^{-1}\right)^{-1} = \gamma^{log_\alpha \beta} \pmod p$$

□

Totuşi, protocolul de bază Diffie - Hellman are o vulnerabilitate majoră: nu rezistă unui atac *man-in-the-middle*.
Schematic, protocolul Diffie - Hellman se efectuează după schema următoare:

$$A \xrightarrow{\alpha^a} \xleftarrow{\alpha^b} B$$

Să presupunem că într-un atac activ, *Oscar* (O) se interpune între A şi B în modul următor:

$$A \xrightarrow{\alpha^a} \xleftarrow{\alpha^{b'}} O \xrightarrow{\alpha^{a'}} \xleftarrow{\alpha^b} B$$

unde $a', b' \in Z_q^*$ sunt generate aleator de O.

O interceptează mesajele lui A şi B şi le înlocuieşte cu ale sale. La sfârşitul protocolului, el a stabilit o cheie de comunicaţie $\alpha^{ab'}$ cu A şi o cheie $\alpha^{a'b}$ cu B.
Când A doreşte să trimită un mesaj lui B, el va utiliza cheia pe care o împarte cu O; acesta poate decripta mesajul şi apoi să îl cripteze cu cheia comună cu B.
B primeşte mesajul, fără să realizeze că a fost citit de O.

Acelaşi lucru se întâmplă în cazul unui mesaj trimis de B către A.

5.5.3 Punerea de acord MTI

Protocolul MTI propus de Matsumoto, Takashima şi Imai, poate fi îmbunătăţit astfel ca să reziste unui atac *man-in-the-middle*.

1. A generează aleator $x \in Z_{p-1}^*$, calculează
$$R_A = \alpha^x \pmod p$$
şi trimite lui B perechea $(Cert(A), R_A)$.
2. B generează aleator $y \in Z_{p-1}^*$, calculează
$$R_B = \alpha^y \pmod p$$
şi trimite lui A perechea $(Cert(B), R_B)$.
3. A calculează $K_{A,B} = R_B^a \cdot Y_B^x \pmod p$, iar B calculează $K_{B,A} = R_A^b \cdot Y_A^y \pmod p$.

Cheia comună de sesiune este deci

$$K = K_{A,B} = K_{B,A} = \alpha^{ay+bx} \pmod p$$

Exemplul 5.3. *Să luăm* $p = 27803$ *şi* $\alpha = 5$.
Dacă A *alege* $a = 21131$, *el va calcula* $Y_A = 5^{21131} = 21420 \pmod{27803}$, *pe care îl pune în certificatul său.*
La fel, dacă B *alege* $b = 17555$, *va avea* $Y_B = 5^{17555} = 17100 \pmod{27803}$

Presupunem că A *selectează* $x = 169$; *el va trimite lui* B

$$R_A = 5^{169} = 6268 \pmod{27803}$$

Dacă B *alege* $y = 23456$, *el trimite lui* A

$$R_B = 5^{23456} = 26759 \pmod{27803}$$

Acum se poate calcula

$$K_{A,B} = R_B^a \cdot Y_B^x \ (mod\ p) = 26759^{21131} \cdot 17100^{169} = 21600 \ (mod\ 27803)$$

Aceeaşi cheie o obţine şi B.

Semnătura arbitrului elimină posibilitatea interpunerii lui *Oscar*.

Într-adevăr, dacă un intrus O se interpune între A şi B, va avea loc scenariul

$$A \ \underset{Cert(B),\alpha^{y'}}{\overset{Cert(A),\alpha^{x}}{\rightleftarrows}} \ O \ \underset{Cert(B),\alpha^{y}}{\overset{Cert(A),\alpha^{x'}}{\rightleftarrows}} \ B$$

În acest moment, A şi B vor calcula chei diferite: A calculează $K = \alpha^{xb+y'a} \ (mod\ p)$, iar B calculează $K = \alpha^{x'b+ya} \ (mod\ p)$.

În plus, nici una din cheile lui A sau B nu poate fi calculată de O, pentru că aceasta ar necesita cunoaşterea exponentului secret "a" respectiv "b". Deci, deşi A şi B obţin chei distincte, nici una din ele nu poate fi calculată şi de O (în ipoteza că problema logaritmului discret este dificilă).

Altfel spus, A şi B sunt siguri că nici o altă persoană nu poate calcula cheia. Avem deci o *autentificare implicită* a cheii de sesiune.

Pentru atacuri pasive, securitatea sistemului MTI se reduce tot la dificultatea problemei Diffie - Hellman.

5.5.4 Protocoale AK înrudite cu protocolul Diffie - Helman

Vom prezenta câteva protocoale de punere de acord a cheilor cu autentificare (AK) utilizate frecvent în standarde criptografice. În toate aceste cazuri, confirmarea cheii este făcută prin protocoale separate, cuprinse în standarde.

KEA

Protocolul KEA (Key Exchange Algorithm) a fost propus de NSA şi declasificat în 1998.

1. A şi B obţin – din certificate – cheile publice Y_B respectiv Y_A.
2. A generează aleator $x \in Z_q^*$ şi trimite lui B valoarea $R_A = \alpha^x$.
3. B generează aleator $y \in Z_q^*$ şi trimite lui A valoarea $R_B = \alpha^y$.

4 A verifică
$$1 < R_B < p, \qquad (R_B)^q \equiv 1 \pmod p$$
Dacă relaţiile sunt îndeplinite, calculează cheia
$$K_{A,B} = (Y_B)^x + (R_B)^a \pmod p$$

5 B procedează similar, obţinând – în caz de succes – cheia
$$K_{B,A} = (Y_A)^y + (R_A)^b \pmod p$$

6 A şi B calculează cheia de sesiune (de 80 biţi) $K = e(K_{A,B}) = e(K_{B,A})$, unde e este funcţia de criptare a sistemului simetric SKIPJACK ([84]).

Din calcule rezultă imediat
$$K_{A,B} = K_{B,A} = \alpha^{ay} + \alpha^{bx}$$

Exemplul 5.4. *Fie $p = 43$ şi $q = 7$ (divizor al lui $p - 1 = 42$). Se alege apoi $\alpha = 4$ (element de ordin 7 în Z_{43}^*).*

Să presupunem că $a = 5$ şi $b = 2$; deci cheile publice sunt $Y_A = 4^5 = 35 \pmod{43}$ respectiv $Y_B = 4^2 = 16 \pmod{43}$.

Protocolul KEA va decurge astfel:

1. *A şi B obţin – din certificate – $Y_B = 16$ respectiv $Y_A = 35$.*
2. *A alege $x = 6$ şi trimite lui B valoarea $R_A = 4^6 = 11 \pmod{43}$.*
3. *B alege $y = 3$ şi trimite lui A valoarea $R_B = 4^3 = 21 \pmod{43}$.*
4. *A verifică condiţiile $1 < R_B < 21 < 43$ şi $R_B^q = 21^7 = 1 \pmod{43}$.*
   Cum ele sunt îndeplinite, A calculează cheia
   $$K_{A,B} = 16^6 + 21^5 = 39 \pmod{43}.$$
5. *B verifică condiţiile $1 < R_A < 11 < 43$ şi $R_A^q = 11^7 = 1 \pmod{43}$.*
   Cum ele sunt îndeplinite, B calculează cheia
   $$K_{B,A} = 35^3 + 11^2 = 39 \pmod{43}.$$

Este foarte important ca A (B) să verifice relaţiile de la pasul 3 (respectiv 4).

În caz contrar, sunt posibile unele atacuri, cum ar fi:

1. Dacă A nu verifică relaţia $(R_B)^q \equiv 1 \ (mod\ p)$.
   Atunci utilizatorul B poate afla informaţii despre cheia secretă a a lui A.
   Să presupunem că $p-1$ are un divizor s de lungime mică (până în 40 biţi) şi fie $\beta \in Z_p^*$ de ordin s. Dacă B trimite $R_B = \beta$, atunci A calculează $K_{A,B} = \alpha^{bx} + \beta^a \ (mod\ p)$ şi $K = e(K_{A,B})$.

   Să presupunem acum că A trimite lui B un mesaj criptat $c = e_K(m)$, unde e_K este funcţia de criptare (cu cheia K) a unui sistem de criptare simetric, iar mesajul m are o structură standard. Pentru fiecare $d \in [0, s-1]$, B va calcula $K'_{B,A} = \alpha^{bx} + \beta^d \ (mod\ p)$ şi $K' = e(K'_{B,A})$, după care va decripta $m' = e^{-1}_{K'}(c)$.
   Dacă m' are structura standard, atunci B trage concluzia că $d = a \ (mod\ s)$, deci obţine informaţie parţială despre a.

   Acest lucru se poate repeta pentru diferiţi factori primi mici ai lui $p-1$.

2. Dacă A nu verifică relaţiile $1 < Y_B < p, \ \ 1 < R_B < p$.
   Atunci *Oscar* certifică $Y_O = 1$ drept cheia sa publică, după care trimite lui B cheia publică temporară R_A a lui A, spunând că este a sa. După ce B replică cu R_B, *Oscar* trimite $R'_B = 1$ lui A, spunând că vine de la B. A calculează $K_{A,B} = \alpha^{bx} + 1$ şi B calculează $K_{B,O} = \alpha^{bx} + 1$. Deci B are – fără să ştie – o cheie de sesiune comună cu A.

Construirea cheii de sesiune prin criptarea cu un sistem de criptare simetric a cheii comune, are cel puţin două argumente în favoarea sa:

i. În acest fel se amestecă biţii "tari" ai sistemului cu eventuali biţi "slabi" – biţi care oferă informaţie despre cheia comună $K_{A,B}$, sau pot fi ghiciţi cu o probabilitate semnificativă.

ii. Se distrug relaţiile algebrice dintre cheia comună şi cheile publice Y_A, Y_B, R_A, R_B. Acest lucru previne un atac cu cheie cunoscută, numit *atacul triunghiular al lui Burmester*, care poate fi descris astfel:

   C este un utilizator cu perechea de chei (c, α^c). El observă o execuţie a protocolului între A şi B, în care aceştia inter-schimbă cheile publice temporare $R_A = \alpha^x, \ R_B = \alpha^y$; cheia comună rezultată este $K_{A,B} = K_{B,A} = \alpha^{ay} + \alpha^{bx}$.
   C iniţiază cu A o execuţie a protocolului, folosind drept cheie publică temporară $R_C = \alpha^y$. Dacă A foloseşte acum $R_A = \alpha^{x'}$, va rezulta o cheie $K_{A,C} = \alpha^{ay} + \alpha^{cx'}$ pe care o poate calcula numai A.
   Similar, C iniţiază cu B o execuţie a protocolului, folosind $R_C = \alpha^x$; rezultă o cheie $K_{B,C} = \alpha^{bx} + \alpha^{cy'}$ care poate fi calculată numai de B.
   Dacă C poate afla prin alte mijloace $K_{A,C}$ şi $K_{B,C}$ (faza de "cheie cunoscută" a atacului), el va putea determina

$$K_{A,B} = K_{A,C} + K_{B,C} - \alpha^{cx'} - \alpha^{cy'}.$$

Modelul unificat

Modelul unificat (The Unified Model) ([6]) este un protocol utilizat în prima versiune a standardelelor ANSI $X9.42$, ANSI $X9.63$ şi IEEE $P1363$. Avantajul său este simplitatea – şi deci uşurinţa analizei sale.

1. A generează aleator $x \in Z_q^*$; trimite lui B elementele $R_A = \alpha^x$ şi $Cert(A)$.
2. B generează aleator $y \in Z_q^*$; trimite lui A elementele $R_B = \alpha^y$ şi $Cert(B)$.
3. A verifică relaţiile
$$1 < R_B < p, \qquad (R_B)^q \equiv 1 \pmod p$$
Dacă ele sunt îndeplinite, A calculează cheia de sesiune
$$K = h((Y_B)^a \,\|\, (R_B)^x)$$
4. B procedează similar. Cheia de sesiune este obţinută de B cu formula
$$K = h((Y_A)^b \,\|\, (R_A)^y)$$

Cheia comună calculată de A şi B este $K = h(\alpha^{ab} \| \alpha^{xy})$.

MQV

Protocolul MQV (Menezes, Qu, Solinas) este inclus în 1998 în versiunile iniţiale ale standardelor ANSI $X9.42$, ANSI $X9.63$ şi IEEE $P1363$. El are multe puncte comune cu *Modelul unificat*, diferenţele apărând în calculul cheii de sesiune.

1. A generează aleator $x \in Z_q^*$; trimite lui B elementele şi $R_A = \alpha^x$ şi $Cert(A)$.
2. B generează aleator $y \in Z_q^*$; trimite lui A elementele şi $R_B = \alpha^y$ şi $Cert(B)$.
3. A verifică relaţiile
$$1 < R_B < p, \qquad (R_B)^q \equiv 1 \pmod p$$
Dacă ele sunt îndeplinite, A calculează
$$s_A = (x + a \cdot \overline{R}_A) \pmod q, \qquad K_A = \left(R_B \cdot (Y_B)^{\overline{R}_B}\right)^{s_A}.$$
Dacă relaţiile nu sunt verificate sau $K_A = 1$, atunci A opreşte protocolul.

4 B procedează similar şi calculează
$$s_B = (y + b \cdot \overline{R}_B) \pmod q, \qquad K_B = \left(R_A \cdot (Y_A)^{\overline{R}_A}\right)^{s_B}.$$

5 Cheia comună de sesiune este $K = h(K_A) = h(K_B) = h(\alpha^{s_A s_B})$.

Raportul tehnic care defineşte protocolul foloseşte o notaţie suplimentară: Dacă $X \in Z_p^*$, atunci $\overline{X} = (X \pmod{2^{80}}) + 2^{80}$. Mai general,

$$\overline{X} = (X \pmod{2^{\lceil f/2 \rceil}}) + 2^{\lceil f/2 \rceil}$$

unde f este lungimea lui q, exprimată în biţi. De remarcat că $\overline{X} \pmod q \neq 0$.

Observaţia 5.4. *Utilizarea lui $\overline{R}_A$ foloseşte doar jumătate din biţii lui R_A; în acest fel numărul de operaţii pentru calculul lui $(Y_A)^{\overline{R}_A}$ se înjumătăţeşte fără a afecta securitatea protocolului.*
În plus, definiţia lui $\overline{R}_A$ asigură $\overline{R}_A \neq 0$, deci contribuţia cheii private "a" la calculul lui s_A nu poate fi ignorată.

Verificarea $K_A = 1$ asigură condiţia ca ordinul lui K_A să fie q.
Asupra protocolului MQV poate fi lansat un atac *on-line*:
Să presupunem că C interceptează cheia R_A trimisă spre B; pe baza ei:

1. C calculează $R_C = R_A \cdot (Y_A)^{\overline{R}_A} \cdot \alpha^{-1}$, $c = \left(\overline{R}_C\right)^{-1}$ şi $Y_C = \alpha^c$.
2. C determină cheia Y_C şi o certifică (lucru posibil, deoarece C cunoaşte cheia secretă c).
3. C transmite lui B elementele R_C şi $Cert(C)$.
4. B răspunde cu R_B, pe care C îl forwardează lui A.

În acest moment, A şi B au o cheie de sesiune comună K, pe care B crede că o împarte cu C (nu cu A).

Variante ale protocoalelor AK într-o singură trecere

Principalul scop al protocoalelor AK într-o trecere este punerea de acord a lui A şi B asupra unei chei de sesiune necesare transmiterii unui singur mesaj – de la A la B; acest lucru presupune evident faptul că A are acces la $Cert(B)$. Astfel de protocoale sunt utile în aplicaţii în care numai unul din utilizatori este on-line.

Toate protocoalele prezentate (KEA, *Modelul unificat*, MQV) pot fi convertite în protocoale cu o singură trecere, prin simpla înlocuire a cheii temporare R_B a lui B cu cheia publică Y_B.
Exemplificăm acest lucru cu transformarea protocolului MQV într-un protocol cu o singură trecere.

1. A generează aleator $x \in Z_q^*$; trimite lui B elementele şi $R_A = \alpha^x$ şi $Cert(A)$.
2. A calculează

$$s_A = (x + a \cdot \overline{R}_A) \pmod q, \qquad K_A = \left(Y_B \cdot (Y_B)^{\overline{Y}_B}\right)^{s_A}$$

   Dacă $K_A = 1$, atunci A opreşte execuţia (cu eşec).
3. B verifică relaţiile

$$1 < R_A < p, \qquad (R_A)^q \equiv 1 \pmod p$$

   Dacă aceste condiţii sunt îndeplinite, atunci B calculează

$$s_B = (b + b \cdot \overline{Y}_B) \pmod q, \qquad K_B = \left(R_A \cdot (Y_A)^{\overline{R}_A}\right)^{s_B}$$

   Dacă $K_B = 1$, atunci B opreşte protocolul (cu eşec).
4. Cheia de sesiune este $K = h(K_A) = h(K_B) = h(\alpha^{s_A s_B})$.

5.5.5 Protocoale AKC înrudite cu protocolul Diffie - Helman

Protocoalele explicite de punere de acord a cheii se obţin de obicei din protocoale implicite însoţite de un mesaj de autentificare (un MAC). În general ele sunt preferate protocoalelor AK, deoarece confirmarea asigură un plus de securitate.

Protocoale AKC derivate din protocoale AK

Vom construi un protocol în trei treceri derivat din *Modelul unificat*, la care se adaugă un MAC care autentifică numărul trecerii, utilizatorii şi cheile temporare.
Se mai folosesc două funcţii de dispersie h_1, h_2. În practică, ele sunt definite

$$h_1(m) = h(10, m), \quad h_2(m) = h(01, m)$$

unde h este o funcţie de dispersie criptografică.

1. A generează aleator $x \in Z_q^*$; trimite lui B elementele $R_A = \alpha^x$ şi $Cert(A)$.
2. (a) B verifică
$$1 < R_A < p, \qquad (R_A)^q \equiv 1 \pmod p$$
Dacă relaţiile nu sunt îndeplinite, B opreşte execuţia (cu eşec).
   (b) B generează $y \in Z_q^*$ şi calculează
$$k' = h_1\left((Y_A)^b \,\|\, (R_A)^y\right), \quad K = h_2\left((Y_A)^b \,\|\, (R_A)^y\right),$$
$$R_B = \alpha^y, \qquad m_B = MAC_{k'}(2, B, A, R_B, R_A).$$
   (c) B trimite lui A tripletul $(Cert(B),\ R_B,\ m_B)$.
3. (a) A verifică
$$1 < R_B < p, \qquad (R_B)^q \equiv 1 \pmod p$$
Dacă relaţiile nu sunt îndeplinite, B opreşte execuţia (cu eşec).
   (b) A calculează
$$k' = h_1\left((Y_B)^a \,\|\, (R_B)^x\right), \quad m'_B = MAC_{k'}(2, B, A, R_B, R_A)$$
şi verifică egalitatea $m'_B = m_B$.
   (c) A calculează
$$m_A = MAC_{k'}(3, A, B, R_A, R_B), \quad K = h_2\left((Y_B)^a \,\|\, (R_B)^x\right)$$
şi trimite lui B valoarea m_A.
4. B calculează $m'_A = MAC_{k'}(3, A, B, R_A, R_B)$ şi verifică egalitatea $m'_A = m_A$.
5. Cheia de sesiune este K.

În mod similar se pot construi protocoale AKC derivate din KEA şi MQV. Variantele AKC ale protocoalelor MQV şi *Model Unificat* sunt incluse în standardul ANSI $X9.63$.

Protocol între staţii

Tot cu scopul de a evita atacurile de tip *man-in-the-middle*, Diffie, Van Oorschot şi Wiener au propus un protocol numit STS (*protocol între staţii*).
Implementat în diverse variante, structura sa de bază este definită astfel:

1. A generează aleator un număr $a \in Z_{p-1}^*$; apoi calculează numărul $\alpha^a \ (mod\ p)$ pe care îl trimite lui B.
2. B generează aleator un număr $b \in Z_{p-1}^*$.
3. B calculează $\alpha^b \ (mod\ p)$, apoi
$$K = (\alpha^a)^b \ (mod\ p), \quad y_B = sig_B(\alpha^b, \alpha^a)$$
Trimite lui A mesajul $(Cert(B), \alpha^b \ (mod\ p), y_B)$.
4. A calculează
$$K = \left(\alpha^b\right)^a \ (mod\ p)$$
şi verifică $Cert(V)$ cu ver_T, apoi y_B cu ver_B extras din $Cert(B)$.
5. A calculează $y_A = sig_A(\alpha^a, \alpha^b)$ şi trimite lui B mesajul $(Cert(A), y_A)$.
6. B verifică $Cert(A)$ cu ver_T, apoi y_A cu ver_A extras din $Cert(A)$.

În această schemă, numărul prim p şi elementul primitiv $\alpha \in Z_p$ sunt publice. Fiecare utilizator A dispune de un protocol privat de semnătură sig_A şi unul public de verificare ver_A.
Arbitrul T are de asemenea asociate funcţiile sig_T respectiv ver_T. Certificatul lui A este

$$Cert(A) = (ID(A), ver_A, sig_T(ID(A), ver_A))$$

Informaţiile schimbate în cadrul protocolului STS simplificat sunt schematizate de diagrama următoare:

$$A \quad \xrightarrow{\alpha^a} \quad \xleftarrow{\alpha^b, sig_B(\alpha^b, \alpha^a)} \quad \xrightarrow{sig_A(\alpha^a, \alpha^b)} \quad B$$

Prin acest protocol, un intrus O nu se poate interpune între A şi B.
Într-adevăr, dacă O interceptează α^a şi îl înlocuieşte cu $\alpha^{a'}$, el va trebui să înlocuiască de asemenea şi $sig_B(\alpha^b, \alpha^{a'})$ cu $sig_B(\alpha^{b'}, \alpha^a)$, ceea ce nu poate decât în cazul $a = a'$ şi $b = b'$ (pentru că nu cunoaşte sig_B).

La fel, O nu poate înlocui $sig_A(\alpha^{a'}, \alpha^b)$ cu $sig_A(\alpha^a, \alpha^{a'})$, pentru că nu cunoaşte sig_A.

Varianta aceasta de protocol nu oferă totuşi o confirmare a cheii. Pentru aceasta trebuie modificat $y_B = e_K(sig_B(\alpha^b, \alpha^a))$ în pasul 3 şi $y_A = e_K(sig_A(\alpha^a, \alpha^b))$ în pasul 5.

În acest fel – ca la *Kerberos* – se obţine o confirmare a cheii decriptând o parte cunoscută a cheii de sesiune. Acesta este protocolul STS complet.

5.6 Chei auto-certificate

Metoda punerii de acord prezentată în acest paragraf este construită de Girault; ea nu necesită certificat. Valoarea cheii publice asigură o autentificare implicită.

Protocolul lui Girault combină proprietăţile sistemului RSA cu cele ale logaritmilor discreţi.

Fie $n = pq$ unde $p = 2p_1 + 1,\ q = 2q_1 + 1$ sunt numere prime tari (p_1, q_1 sunt numere prime distincte).
Cum grupul multiplicativ Z_n^* este izomorf cu $Z_p \times Z_q$, ordinul maxim al unui element din Z_n^* este $cmmmc(p-1, q-1) = 2p_1q_1$.

Fie $\alpha \in Z_n^*$ de ordin $2p_1q_1$. Vom utiliza problema logaritmului discret în subgrupul ciclic al lui Z_n^* generat de α.

În acest protocol, factorizarea $n = pq$ este cunoscută numai de către arbitrul T. Valorile n, α sunt publice, iar p, q sunt secrete (deci şi p_1, q_1).

T alege un exponent de criptare RSA public, să spunem e. Exponentul de decriptare $d = e^{-1}\ (mod\ \phi(n))$ este secret.

Fiecare utilizator A are un identificator $ID(A)$ şi primeşte de la T o cheie publică auto-certificată R_A, conform următorului protocol:

1. A alege un exponent secret a şi calculează $Y_A = \alpha^a\ (mod\ n)$.
2. A trimite lui T valorile (a, Y_A).
3. T calculează $R_A = (Y_A - ID(A))^d\ (mod\ n)$, pe care îl trimite lui A.

De remarcat aportul arbitrului în calculul lui R_A. Se observă că

$$Y_A = R_A^e + ID(A) \quad (mod\ n)$$

poate fi calculat folosind numai informaţiile publice.

Protocolul lui Girault este dat mai jos. Schematizat, schimbul de informaţii arată în felul următor:

$$A \quad \xrightarrow{\quad ID(A), R_A, \alpha^x\ (mod\ n) \quad} \quad B$$
$$A \quad \xleftarrow{\quad ID(B), R_B, \alpha^y\ (mod\ n) \quad} \quad B$$

La sfârşitul acestui protocol, A şi B dispun de aceeaşi cheie:

$$K = \alpha^{bx+ay} \quad (mod\ n)$$

1. A generează aleator x şi calculează $S_A = \alpha^x \ (mod\ n)$;
   tripletul $(ID(A), R_A, S_A)$ este trimis lui B.

2. B alege aleator y şi calculează $S_B = \alpha^y \ (mod\ n)$;
   tripletul $(ID(B), R_B, S_B)$ este trimis lui A.

3. A calculează cheia

$$K_{A,B} = S_B^a \cdot (R_B^e + ID(B))^x \quad (mod\ n).$$

   Cheia calculată de B este

$$K_{B,A} = S_A^b \cdot (R_A^e + ID(A))^y \ (mod\ n)$$

Exemplul 5.5. *Să presupunem* $p = 839$ *şi* $q = 863$. *Vom avea* $n = 724057$ *şi* $\phi(n) = 722356$. *Elementul* $\alpha = 5$ *are ordinul* $2p_1q_1 = \phi(n)/2$.

Dacă arbitrul T *alege* $e = 84453$ *drept exponent de criptare, vom avea* $d = 125777$.

Dacă $ID(A) = 500021$ *şi* $a = 111899$, *vom avea* $Y_A = 488889$ *şi* $R_A = 650704$.
În mod similar, considerăm $ID(B) = 500022$ *şi* $b = 123456$, *deci* $Y_B = 111692$, $R_B = 683556$.

Dacă A *şi* B *vor să stabilească o cheie de sesiune şi* A *alege numărul* $x = 56381$, *iar* B *numărul* $y = 356935$, *vom avea* $S_A = 171007$, $S_B = 320688$.

După protocol, cei doi utilizaotri vor dispune de cheia comună $K = 42869$.

5.6.1 Securitatea sistemului cu chei auto-certificate

Cum valorile $Y_A, R_A, ID(A)$ nu sunt semnate de autoritatea T, nimeni nu poate verifica direct autenticitatea lor.

Să presupunem că ele provin de la O (fără ajutorul arbitrului), care vrea să se dea drept A.

Dacă O furnizează $ID(A)$ şi dacă R_A conduce la un Y'_A greşit, nu se poate calcula exponentul a' asociat (dacă problema logaritmului discret este dificilă).
Fără a', O nu poate determina cheia.

O situaţie similară apare dacă O se interpune între A şi B.
El poate împiedica pe A şi B să obţină o cheie comună, dar nu poate efectua calculele lor. Are loc deci o autentificare implicită.

O întrebare ar fi: *De ce* A *trebuie să comunice arbitrului valoarea* a *?*
T poate determina R_A plecând de la Y_A, fără să cunoască a. Acest lucru se face pentru ca arbitrul să fie convins – înainte de a calcula R_A – că A posedă cheia secretă a.

Dacă T nu face această verificare înainte de calculul lui R_A, sistemul poate fi atacat.

Să presupunem că O alege un a' fals şi determină $Y'_A = \alpha^{a'} \ (mod\ n)$.

Cu aceste elemente, el stabileşte o cheie publică falsă $R'_A = (Y'_A - ID(A))^d \ (mod\ n)$ în felul următor:

O calculează $Y'_O = Y'_A - ID(A) + ID(O)$ şi trimite lui T perechea $(Y'_O, ID(O))$.

Presupunem că arbitrul calculează efectiv pentru O valoarea

$$R'_O = (Y'_O - ID(O))^d \ (mod\ n).$$

Atunci, din

$$Y'_O - ID(O) \equiv Y'_A - ID(A) \quad (mod\ n)$$

se obţine imediat $R'_O = R'_A$.

Să presupunem acum că A şi B efectuează protocolul, iar O se interpune conform schemei următoare:

$$A \xrightarrow[\xleftarrow{ID(B), R_B, \alpha^y \ (mod\ n)}]{ID(A), R_A, \alpha^x \ (mod\ n)} O \xrightarrow[\xleftarrow{ID(B), R_B, \alpha^y \ (mod\ n)}]{ID(A), R'_A, \alpha^{x'} \ (mod\ n)} B$$

B calculează deci cheia $K' = \alpha^{x'b+ya'} \ (mod\ n)$, iar A calculează $K = \alpha^{xb+ya} \ (mod\ n)$.

O obţine K' calculând $K' = S_B^{a'} \cdot (R_B^e + ID(B)))^{x'} \ (mod\ n)$.

O şi B posedă deci aceeaşi cheie, în timp ce B crede că o împarte cu A.

În acest moment, O poate decripta mesajele trimise de B pentru A.

5.7 Exerciţii

5.1. *Considerăm protocolul Blom (cazul $k = 1$) pentru utilizatorii A, B, C, D. Se dau valorile*

$$p = 7873, \quad r_A = 2365, \quad r_B = 6648, \quad r_C = 1837, \quad r_D = 2186.$$

Polinoamele secrete sunt:

$$g_A(x) = 6018 + 6351x, \quad g_B(x) = 3749 + 7121x,$$
$$g_C(x) = 7601 + 7802x, \quad g_D(x) = 635 + 6828x.$$

1. *Calculaţi cheile de sesiune pentru fiecare pereche de utilizatori (şi verificaţi de exemplu că $K_{A,B} = K_{B,A}$).*

2. *Arătaţi cum C şi D pot calcula $K_{A,B}$.*

5.2. *Considerăm protocolul Blom având $k = 2$, construit pentru 5 utilizatori: A, B, C, D, E. Datele iniţiale sunt*

$$p = 97, \quad r_A = 14, \quad r_B = 38, \quad r_C = 92, \quad r_D = 69, \quad r_E = 70.$$

Polinoamele secrete sunt

$$g_A(x) = 15 + 15x + 2x^2, \quad g_B(x) = 95 + 77x + 83x^2, \quad g_C(x) = 88 + 32x + 18x^2,$$
$$g_D(x) = 62 + 91x + 59x^2, \quad g_E(x) = 10 + 82x + 52x^2.$$

1. *Calculaţi cheia de sesiune* $K_{A,B} = K_{B,A}$.

2. *Arătaţi cum calculează alianţa* C, D, E *cheia* $K_{A,B}$.

5.3. *În protocolul Diffie - Hellman de perioadă scurtă avem* $p = 11$ *şi* $\alpha = 2$.

1. *Dacă* $R_A = 9$, *cât este a ?*

2. *Dacă cheia de sesiune este* $K = 3$, *cât a fost* R_B *?*

5.4. *În protocolul Diffie - Hellman avem* $p = 27001$, $\alpha = 101$. *Dacă* $a = 21768$ *şi* $b = 9898$, *determinaţi cheia de sesiune* $K_{A,B}$.

5.5. *Să presupunem că* A, B, C *doresc să stabilească o cheie comună de sesiune folosind o variantă a protocolului Diffie - Hellman. Cheia pe care o doresc este de forma*

$$\alpha^{abc} \ (mod\ p)$$

unde a *este ales de* A, b *este ales de* B *şi* c *este ales de* C.
Efectuaţi calculele parcurgând toate cele şase comunicări.

5.6. *Fie următoarea generalizare a protocolului Diffie - Hellman pentru stabilirea unei chei de sesiune între* $t \geq 2$ *utilizatori* $A_0, \ldots A_{t-1}$*:*

1. *Fiecare* A_i *generează aleator* $x_i \in Z_{p-1}^*$ *şi trimite* $y_i = \alpha^{x_i} \ (mod\ p)$ *la ceilalţi* $t-1$ *utilizatori.*

2. *Fiecare* A_i *calculează* $z_i = \left(y_{i+1} \cdot y_{i-1}^{-1}\right)^{x_i} \ (mod\ p)$ *şi trimite* z_i *la ceilalţi* $t-1$ *utilizatori.*

3. *Fiecare* A_i *calculează cheia de sesiune*

$$K = y_{i-1}^{tx_i} \cdot z_i^{t-1} z_{i+1}^{t-2} \ldots z_{i+t-3}^2 \cdot z_{i+t-2} \quad (mod\ p)$$

(toţi indicii de la paşii 2 *şi* 3 *sunt calculaţi modulo* t*).*

1. *Arătaţi că la sfârşitul protocolului toţi utilizatorii au aceeaşi cheie.*

2. *Arătaţi că pentru un adversar pasiv, a afla cheia de sesiune este echivalent cu a rezolva problema Diffie - hellman.*

5.7. *A şi B efectuează protocolul MTI cu* $p = 30113$ *şi* $\alpha = 52$. *Presupunem că A are cheia secretă* $a = 8642$ *şi alege* $x = 28654$, *iar B are cheia* $b = 24673$ *şi alege* $y = 12385$. *Verificaţi paşii protocolului şi determinaţi cheia de sesiune* $K_{A,B} = K_{B,A}$.

5.8. *Dacă un adversar pasiv încearcă să calculeze cheia K construită de A şi B pe baza protocolului MTI, el trebuie să rezolve o apariţie a problemei MTI reprezentată sub forma*

Enunţ: *Fie* $I = (p, \alpha, \beta, \gamma, \delta, \epsilon)$, *unde p este număr prim,* $\alpha \in Z_p$ *este primitiv, şi* $\beta, \gamma, \delta, \epsilon \in Z_p^*$.
Cerinţă: *Să se determine* $\beta^{log_\alpha \gamma} \delta^{log_\alpha \epsilon} \pmod p$.

Să se arate că un oracol care rezolvă problema MTI poate rezolva problema Diffie - Hellman şi reciproc.

5.9. *Se consideră protocolul Girault cu* $p = 167$, $q = 179$; *deci* $n = 29893$. *Presupunem* $\alpha = 2$ *şi* $e = 11101$.

1. *Calculaţi exponentul de decriptare d.*
2. *Ştiind* $ID(A) = 10021$ *şi* $a = 9843$, *calculaţi* Y_A *şi* R_A.
   Similar, ştiind $ID(B) = 10022$, *calculaţi* Y_B, R_B.
3. *Verificaţi cum se poate determina* Y_A *folosind* $R_A, ID(A)$ *şi e.*
   Similar pentru Y_B.
4. *Presupunem că A alege* $x = 15556$, *iar B alege* $y = 6420$. *Calculaţi* S_A, S_B *şi arătaţi cum A şi B obţin o cheie de sesiune comună.*

Capitolul 6

IBE (Identity - Based Encryption)

Stabilirea cheilor unui sistem de criptare cu cheie publică constituie un proces destul de dificil. Astfel

- Utilizatorul – sau un agent care operează în numele său – generează o pereche de chei (public, privat), pe care o autoritate de certificare (CA) o semnează, o completează cu alte informaţii, creind un certificat pe care apoi îl gestionează.
  În plus, pentru a evita pierderea sau a răspunde diverselor solicitări de autentificare, acest certificat este arhivat într-un "registru".

- Procesul de generare a cheii este destul de dificil computaţional. Generarea a două numere de 512 cifre binare, verificarea că ele sunt prime şi înmulţirea lor pentru a forma o cheie RSA de 1024 biţi este momentan o problemă relativ uşoară; dar în viitorul destul de apropiat - când lungimea cheilor va fi creşte din necesităţi de securitate - va crea probleme de complexitate pentru un calculator de putere medie.

Din aceste motice, o cheie publică are o perioadă de valabilitate relativ lungă: de obicei între 1 şi 3 ani.

Pentru a utiliza o cheie publică aflată într-un certificat digital, un utilizator trebuie să o caute într-un registru, să găsească certificatul care o conţine, să verifice dacă este valid (lucru care se realizează sau căutând într-o listă de certificate invalide, sau apelând la un serviciu on-line care îi returnează starea de validitate a certificatului). Apoi să extragă cheia din certificat şi să o folosească la criptarea unui mesaj. Procesul este destul de costisitor, mai ales pentru cazurile când gradul de secretizare al mesajelor nu este prea ridicat.

În 1984 Adi Shamir ([72]) sugerează o modalitate ceva mai simplă de operare, care să evite dificultăţile menţionate mai sus.

Într-un astfel de sistem, utilizatorul *Bob* generează o secvenţă binară care constituie identitatea sa (ID). Pentru a evita ca acest ID să fie utilizat prea frecvent, de obicei el conţine şi alte informaţii, cum ar fi o perioadă de valabilitate a cheii, sau un server de

utilizare. De aici vine numele de *Identity - Based Encryption* (IBE).
Restul componentelor din ID sunt oferite de:

1. O autoritate de încredere, numită *Trusted Third Party* (TTP) care generează componentele schemei ce va fi folosită;

2. Un generator de chei private (PKG), care calculează o cheie privată folosind componentele secrete şi ID-ul dat de *Bob*. Această cheie este livrată lui *Bob* printr-un un canal securizat.

O schemă IBE conţine 4 algoritmi:

1. Alegerea parametrilor de securitate (Setup). Aici TTP iniţializează toate componentele sistemului (publice şi secrete).

2. Calculul cheii private (Extraction) de către PKG, pe baza componentei secrete a schemei şi a ID-ului dat de utilizator.

3. Algoritmul de criptare a unui text clar (privit ca o secvenţă binară), bazat pe ID şi pe componentele publice ale schemei;

4. Algoritmul de decriptare a mesajului primit, bazat pe ID şi pe cheia privată.

În general, o schemă IBE asigură doar confidenţialitatea mesajului, nu şi celelalte proprietăţi ale unui sistem de criptare (integritate, autenticitate, non-repudiere). Din acest motiv, el este adesea însoţit şi de o schemă de semnătură digitală.

Cele mai cunsocute scheme IBE sunt: schema Cocks (aparent primul astfel de protocol), schemele Boneh - Franklin (cele mai cunoscute şi utilizate) şi schema Sakai - Kasahara. Cu excepţia primeia, toate celelalte au ca bază de lucru criptografia pe curbe eliptice în general, perechile biliniare Tate în particular.

6.1 Algoritmul Cocks

Schema IBE propusă de Cocks în [21] se bazează două probleme dificile: problema factorizării şi problema resturilor pătratice (a se vedea [2]). Ideea este de a cripta fiecare bit al textului clar într-o pereche de numere întregi modulo un număr neprim, suficient de mare pentru ca factorizarea sa să fie dificilă.

6.1.1 Alegerea parametrilor de securitate

Se aleg:

- Două numere prime $p, q, \quad p \equiv 3 \ (mod\ 4), \quad q \equiv 3 \ (mod\ 4)$
- $n = p \cdot q$.
  n este public, iar p, q sunt secrete.
- O funcţie de dispersie $H_1 : \{0,1\}^* \longrightarrow Z_n$, cu restricţia: pentru datele ID ale utilizatorului, dacă $H_1(ID) = a$ atunci simbolul Jacobi

$$\left(\frac{a}{n}\right) = +1$$

  Valoarea a este cheia publică corespunzătoare identităţii ID.

Această restricţie se poate realiza – de exemplu – folosind o funcţie de dispersie oarecare H, determinând $H(ID) = a$ şi apoi incrementând (eventual) a până se ajunge la $\left(\frac{a}{n}\right) = +1$.

Observaţia 6.1. *Relaţia* $\left(\frac{a}{n}\right) = +1$ *arată că* a *este rest pătratic modulo* n. *Din*

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p}\right) \cdot \left(\frac{a}{q}\right)$$

rezultă că simbolurile Legendre $\left(\frac{a}{p}\right)$ *şi* $\left(\frac{a}{q}\right)$ *au acelaşi semn.*
Deoarece $p \equiv 3 \ (mod\ 4), \quad q \equiv 3 \ (mod\ 4)$, *vom avea*

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

deci
$\left(\frac{-a}{n}\right) = \left(\frac{-a}{p}\right) \cdot \left(\frac{-a}{q}\right) = \left(\frac{a}{p}\right) \cdot \left(\frac{-1}{p}\right) \cdot \left(\frac{a}{q}\right) \cdot \left(\frac{-1}{q}\right) = \left(\frac{a}{p}\right) \cdot (-1) \left(\frac{a}{q}\right) \cdot (-1) = \left(\frac{a}{p}\right) \cdot \left(\frac{a}{q}\right) = +1$

Ca o concluzie, nu se ştie apriori care din valorile a *sau* $-a$ *este rest pătratic. Această ambiguitate crează unele dificultăţi în utilizarea schemei Cocks pentru criptare, deoarece duce la dublarea lungimii textului criptat.*

Simbolul Jacobi $\left(\frac{a}{n}\right)$ poate fi calculat direct, fără a cunoaşte factorizarea lui n. Pentru aceasta se poate folosi algoritmul ([53], pag. 25):

$SimbolJacoby(n, a)$:
Intrare: $n \geq 3$ întreg impar, $a \in [0, n)$ întreg.
Ieşire: Simbolul Jacobi $\left(\frac{a}{n}\right)$.

1. **Dacă** $a \leq 1$ **atunci output** a, Stop.
2. $a = 2^m a_1$, unde a_1 este impar.
3. **Dacă** m este par, **atunci** $s \longleftarrow 1$
   altfel dacă $n \equiv \pm 1 \ (mod\ 8)$ **atunci** $s \longleftarrow 1$
   altfel $s \longleftarrow -1$;
4. $n_1 \longleftarrow n \ (mod\ a_1)$;
5. **Output** $s \cdot SimbolJacobi(n_1, a_1)$.

6.2 Descrierea schemei Cocks

6.2.1 Calculul cheii private

Generatorul de chei PKG (care cunoaşte factorizarea lui n) va calcula cheia privată corespunzătoare cheii publice a după formula

$$r = a^{(\phi(n)+4)/8} \ (mod\ n)$$

Reamintim, $\phi(n)$ este numărul de numere întregi prime cu n din intervalul $[1, n-1]$. Dacă $n = p \cdot q$ cu p, q numere prime, atunci $\phi(n) = (p-1) \cdot (q-1)$.

Lema 6.1. *Calculul cheii private r este posibil.*

Demonstraţie. Deoarece p şi q sunt congruente cu 3 modulo 4, vom avea
$p - 1 \equiv 2 \ (mod\ 4)$, $q - 1 \equiv 2 \ (mod\ 4)$. Deci $p = 4k_1 + 2$, $q = 4k_2 + 2$. Atunci

$$\phi(n) + 4 = (p-1) \cdot (q-1) + 4 = 8 \cdot (2k_1 \cdot k_2 + k_1 + k_2 + 1)$$

deci $8 | (\phi(n) + 4)$. □

Lema 6.2. *r este o rădăcină pătrată a lui a modulo n.*

Demonstraţie. Folosind teorema lui Euler, avem

$$r^2 = a^{(\phi(n)+4)/4} = \left(a^{\phi(n)}\right)^{1/4} \cdot a \equiv \pm a \ (mod\ n)$$

Dacă a este rest pătratic modulo n, atunci $r^2 \equiv a \ (mod\ n)$, iar dacă $-a$ este rest pătratic modulo n, atunci $r^2 \equiv -a \ (mod\ n)$.

În ambele cazuri, r acţionează ca o cheie privată, corespunzătoare cheii publice a. □

6.2.2 Algoritmul de criptare

Schema Cocks criptează fiecare bit m al textului clar ca o pereche de numere întregi. Anume:

1. Codifică m sub forma $x = (-1)^m$;
2. Generează aleator două numere t_1, t_2 astfel încât
$$\left(\frac{t_1}{n}\right) = \left(\frac{t_2}{n}\right) = x$$
3. Calculează
$$s_1 = \left(t_1 + \frac{a}{t_1}\right) \pmod n$$
4. Calculează
$$s_2 = \left(t_2 - \frac{a}{t_2}\right) \pmod n$$
5. Scoate ca rezultat final (s_1, s_2).

Sunt necesare două numere pentru că expeditorul (*Alice*) nu ştie care valoare (a sau $-a$) este rest pătratic modulo n. În schimb destinatarul (*Bob*) poate calcula imediat r^2 şi verifica dacă $r^2 \equiv a \pmod n$ sau $r^2 \equiv -a \pmod n$; în funcţie de acest rezultat, el va folosi pentru decriptare pe s_1 sau pe s_2.

Observaţia 6.2. *Dacă s-ar folosi o singură valoare t generată aleator (în loc de două) pentru a calcula*

$$s_1 = \left(t + \frac{a}{t}\right) \pmod n, \qquad s_2 = \left(t - \frac{a}{t}\right) \pmod n$$

atunci un criptanalist poate calcula

$$\frac{s_1 + s_2}{t} = t \pmod n$$

şi apoi

$$x = \left(\frac{t}{n}\right)$$

cu ajutorul căruia va afla imediat bitul de text clar m.

6.2.3 Algoritmul de decriptare

La primirea textului criptat (s_1, s_2) şi cunoscând n (public) precum componentele secrete p, q, r, destinatarul va decripta după următorul algoritm:

1. Se calculează r^2;
2. **Dacă** $r^2 \equiv a \ (mod\ n)$ **atunci** $s \longleftarrow s_1$ **altfel** $s \longleftarrow s_2$;
3. Se află $x = \left(\frac{s+2r}{n}\right)$;
4. **Dacă** $x = -1$ **atunci** $m \longleftarrow 0$ **altfel** $m \longleftarrow 1$.

Lema 6.3. *Algoritmul de decriptare este corect.*

Demonstraţie. Dacă $r^2 \equiv a \ (mod\ n)$ atunci

$$s+2r = \left(t_1 - \frac{a}{t_1}\right)+2r = t_1 \cdot \left(1 + \frac{2r}{t_1} - \frac{a}{t_1^2}\right) \equiv t_1 \cdot \left(1 + \frac{2r}{t_1} + \frac{r^2}{t_1^2}\right) \equiv t_1 \left(1 + \frac{r}{t_1}\right)^2 \ (mod\ n)$$

Deci $s + 2r$ şi t_1 sunt (sau nu sunt) simultan pătrate modulo n. De aici rezultă

$$\left(\frac{s+2r}{n}\right) = \left(\frac{t_1}{n}\right) = x$$

din care se poate afla bitul m de text clar.

Dacă $r^2 \equiv -1 \ (mod\ n)$, atunci

$$s+2r = \left(t_2 + \frac{a}{t_2}\right)+2r = t_2 \cdot \left(1 + \frac{2r}{t_2} + \frac{a}{t_2^2}\right) \equiv t_2 \cdot \left(1 + \frac{2r}{t_2} + \frac{r^2}{t_2^2}\right) \equiv t_2 \left(1 + \frac{r}{t_2}\right)^2 \ (mod\ n)$$

şi are loc relaţia

$$\left(\frac{s+2r}{n}\right) = \left(\frac{t_2}{n}\right) = x.$$

□

Exemplul 6.1. *Să considerăm $p = 7$, $q = 11$, deci $n = 77$. Dacă se ia cheia publică $a = 9$, se găseşte cheia privată $r = 25$. Se verifică congruenţa $r^2 \equiv a \ (mod\ n)$.*

Pentru criptarea bitului 0, Alice va codifica pe 0 ca $x = +1$ şi va căuta aleator două numere t astfel ca

$$\left(\frac{t}{77}\right) = +1$$

Să presupunem că numerele găsite cu această proprietate sunt $t_1 = 4$, $t_2 = 6$.

La pasul următor se află cele două valori (s_1, s_2) date de

$$s_1 = \left(t_1 + \frac{a}{t_1}\right) = \left(4 + \frac{9}{4}\right) = 64 \ (mod\ 77)$$

$$s_2 = \left(t_2 - \frac{a}{t_2}\right) = \left(6 - \frac{9}{6}\right) = 43 \ (mod\ 77)$$

Deci $(64, 43)$ *constituie criptarea bitului de text clar* 0.

Bob ştie că cheia sa privată r *verifică relaţia* $r^2 \equiv a \pmod n$*; deci va alege pentru decriptare pe* s_1*. El va calcula*

$$\left(\frac{s_1 + 2r}{n}\right) = \left(\frac{64 + 50}{77}\right) = \left(\frac{114}{77}\right) = +1$$

de unde deduce pe 0 *ca bit de text clar.*

Exemplul 6.2. *Fie valorile de plecare* $p = 7,\ q = 11$ *(deci* $n = 77$*) şi* $a = 10$*.*
Se obţine $r = 23$ *pentru cheia privată, care verifică relaţia (cunoscută de destinatar)* $r^2 \equiv -a \pmod n$*.*

Pentru criptarea bitului 1*, Alice îl codifică întîi ca* $x = -1$*, apoi alege aleator* $t_1 = 8,\ t_2 = 2$ *care satisfac condiţia*

$$\left(\frac{t}{77}\right) = -1$$

Acum expeditorul calculează cele două valori

$$s_1 = \left(t_1 + \frac{a}{t_1}\right) = \left(8 + \frac{10}{8}\right) = 67 \ (mod\ 77)$$

$$s_2 = \left(t_2 - \frac{a}{t_2}\right) = \left(2 - \frac{10}{2}\right) = 74 \ (mod\ 77)$$

şi trimite textul criptat $(67, 74)$*.*

La recepţie, Bob va folosi pentru decriptare numai pe s_2*. El va calcula*

$$\left(\frac{s_2 + 2r}{n}\right) = \left(\frac{74 + 46}{77}\right) = \left(\frac{120}{77}\right) = -1$$

pe care îl decodifică în 1*.*

Observaţia 6.3. *O condiţie necesară ca schema Cocks să funcţioneze este*

$$(s + 2r, n) = 1$$

De exemplu, pentru $p = 7,\ q = 11,\ a = 10$ *se obţine* $r = 23$*. Se doreşte criptarea bitului* 1 *şi se ia aleator* $t_1 = 12,\ t_2 = 5$*.*
Prin calcul se ajunge la perechea criptată $(s_1, s_2) = (0, 3)$*.*
La decriptare, Bob va urma secvenţa de calcul

$$\left(\frac{s_2 + 2r}{n}\right) = \left(\frac{3 + 46}{77}\right) = \left(\frac{49}{77}\right) = 0$$

care nu este o valoare posibilă pentru x*.*

În general sunt $(p-1)+(q-1)-1$ *astfel de situaţii nedorite pentru un modul* $n = pq$*. Deci probabilitatea de eşec a schemei Cocks este*

$$\frac{p + q - 3}{n} \approx \frac{1}{p} + \frac{1}{q}$$

considerată neglijabilă pentru valori mari ale parametrilor p *şi* q*.*

6.2.4 Securitatea schemei Cocks

Evident, dacă modulul n se poate factoriza, se calculează imediat $\phi(n)$ şi apoi r; deci din acest punct de vedere, securitatea schemei Cocks IBE este legată de problema factorizării. Nu este însă singura legătură.

Teorema 6.1. *Securitatea schemei Cocks IBE depinde de problema resturilor pătratice.*

Demonstraţie. Deoarece

$$\left(\frac{1}{n}\right) = \left(\frac{t}{n}\right)\left(\frac{1/t}{n}\right) = +1$$

rezultă

$$\left(\frac{t}{n}\right) = \left(\frac{1/t}{n}\right)$$

şi deci

$$\left(\frac{a/t}{n}\right) = \left(\frac{a}{n}\right)\left(\frac{1/t}{n}\right) = \left(\frac{a}{n}\right)\left(\frac{t}{n}\right)$$

Fie sistemele de congruenţe

$$(1): \begin{cases} t_1 \equiv t \pmod p \\ t_1 \equiv t \pmod q \end{cases} \qquad (2): \begin{cases} t_2 \equiv t \pmod p \\ t_2 \equiv (a/t) \pmod q \end{cases}$$

$$(3): \begin{cases} t_3 \equiv (a/t) \pmod p \\ t_3 \equiv t \pmod q \end{cases} \qquad (4): \begin{cases} t_4 \equiv (a/t) \pmod p \\ t_4 \equiv (a/t) \pmod q \end{cases}$$

Soluţiile lor sunt date de teorema chineză a resturilor:

$$\begin{aligned} t_1 &= t \cdot e_1 + t \cdot e_2 & t_2 &= t \cdot e_1 + (a/t) \cdot e_2 \\ t_3 &= (a/t) \cdot e_1 + t \cdot e_2 & t_4 &= (a/t) \cdot e_1 + (a/t) \cdot e_2 \end{aligned}$$

unde $e_1,\ e_2$ sunt definite de

$$e_1 = \begin{cases} 1 \pmod p \\ 0 \pmod q \end{cases} \qquad e_2 = \begin{cases} 0 \pmod p \\ 1 \pmod q \end{cases}$$

Soluţiile sistemelor $(1) - (4)$ au proprietăţile (deduse foarte uşor):

$$\begin{aligned} &\left(\frac{t_1}{n}\right) = \left(\frac{t}{p}\right)\left(\frac{t}{q}\right) \\ &\left(\frac{t_2}{n}\right) = \left(\frac{t}{p}\right)\left(\frac{a/t}{q}\right) = \left(\frac{t}{p}\right)\left(\frac{a}{q}\right)\left(\frac{t}{q}\right) \\ &\left(\frac{t_3}{n}\right) = \left(\frac{a/t}{p}\right)\left(\frac{t}{q}\right) = \left(\frac{a}{p}\right)\left(\frac{t}{p}\right)\left(\frac{t}{q}\right) \\ &\left(\frac{t_4}{n}\right) = \left(\frac{a/t}{p}\right)\left(\frac{a/t}{q}\right) = \left(\frac{a}{p}\right)\left(\frac{t}{p}\right)\left(\frac{a}{q}\right)\left(\frac{t}{q}\right) \end{aligned} \qquad (A)$$

Dacă a este un pătrat perfect, atunci

$$\left(\frac{a}{p}\right) = \left(\frac{a}{q}\right) = +1$$

deci

$$\left(\frac{t_1}{n}\right) = \left(\frac{t_2}{n}\right) = \left(\frac{t_3}{n}\right) = \left(\frac{t_4}{n}\right)$$

Dar, dacă a nu este pătrat, avem

$$\left(\frac{a}{p}\right) = \left(\frac{a}{q}\right) = -1$$

de unde se deduce

$$\left(\frac{t_1}{n}\right) = -\left(\frac{t_2}{n}\right) = -\left(\frac{t_3}{n}\right) = \left(\frac{t_4}{n}\right)$$

Remarcăm că pentru orice valoare din mulţimea $\{t_1, t_2, t_3, t_4\}$ folosită la intrare, schema Cocks generează acelaşi text criptat. De exemplu, *Alice* poate calcula

$$s = \left(t_1 + \frac{a}{t_1}\right) = t \cdot e_1 + t \cdot e_2 + \frac{a}{t \cdot e_1 + t \cdot e_2} = t + \frac{a}{t}$$

sau

$$s = \left(t_2 + \frac{a}{t_2}\right) = t \cdot e_1 + \frac{a}{t} \cdot e_2 + \frac{a}{t \cdot e_1 + \frac{a}{t} \cdot e_2} = t + \frac{a}{t}$$

Deci, dacă a nu este pătrat perfect, sunt cazuri când texte clare distincte duc la acelaşi text criptat; singurul mod de a le deosebi revine la a determina dacă a este rest pătratic modulo n – adică problema resturilor pătratice. □

Deoarece schema Cocks criptează un singur bit, ea este vulnerabilă la un atac cu text clar ales adaptat. Astfel, să presupunem că *Oscar* dispune de textul clar (pe biţi) $(m_1, m_2, \ldots, m_k)$ împreună cu textul criptat corespunzător $(c_1, c_2, \ldots, c_k)$ şi vrea să afle textul clar asociat textului criptat $(y_1, y_2, \ldots, y_k)$. El va trimite lui *Bob* mesajul $(y_1, c_2, \ldots, c_k)$ şi va aşteptă reacţia acestuia. De exemplu, dacă *Bob* foloseşte acest text criptat ca un secret partajat pentru a crea o cheie de sesiune, *Oscar* va verifica dacă *Bob* a generat aceeaşi cheie din $(y_1, c_2, \ldots, c_k)$ cum a făcut din $(c_1, c_2, \ldots, c_k)$. În caz afirmativ, deduce că y_1 se decriptează în m_1; altfel va decripta y_1 în $1 - m_1$. *Oscar* repetă acest procedeu pentru fiecare componentă a mesajului criptat, găsind treptat toţi biţii textului clar.

Problema principală a schemei Cocks constă însă în mărimea parametrilor. Astfel, dacă dorim o securitate similară unui sistem de criptare care foloseşte o cheie de 128 biţi, schema Cocks o va asigura cu un modul de 3.072 biţi. Deci, o transmitere a unei chei de 128 biţi cu un sistem Cocks va necesita transmiterea a $128 \cdot 2 \cdot 3.072 = 768.432$ biţi. Un număr enorm pentru a garanta securitatea unui sistem de criptare la nivel actual.

6.3 Algoritmul Boneh - Franklin

Algoritmul Boneh - Franklin este prima schemă[1] de tip IBE construită, a cărei securitate a fost demonstrată formal. Varianta de bază foloseşte un secret partajat care poate fi calculat atât de Alice cât şi de Bob. Deoarece nu asigură o securitate completă (nu rezistă la un atac cu text clar ales), ea este completată ulterior de o schemă IBE Boneh-Franklin generalizată.

6.3.1 Schema Boneh - Franklin simplă

Alegerea parametrilor de securitate

Fie

- E o curbă eliptică cu grad de acoperire k peste corpul Z_q (q număr prim);
- Un număr prim p astfel ca $p|\ card(E(Z_q)),\ \ p^2 \not| \ card(E(Z_q))$;
- G_T – grup construit în felul următor:
  Fie $P \in E(Z_q)[p]$ un punct arbitrar şi $G_1 =< P >$ un grup ciclic de ordin p. Definim o pereche Tate
  $$\hat{e} : G_1 \times G_1 \longrightarrow GF(q^k)^*$$
  şi considerăm grupul ciclic $G_T =< \hat{e}(P, P) >$ de ordin p.
- n – număr întreg fixat (lungimea unui bloc de mesaj);
- Două funcţii de dispersie
  $$H_1 : \{0,1\}^* \longrightarrow G_1, \qquad H_2 : G_T \longrightarrow \{0,1\}^n$$
- Un punct arbitrar $Q \in G_1$.

Toate aceste elemente
$$(q, p, n, E, P, Q, G_1, G_T, \hat{e}, H_1, H_2)$$
formează componenta publică a sistemului.

Componenta secretă este numărul s scris pe n biţi, cu proprietatea $Q = sP$ (altfel spus: $s = log_P Q$).

Observaţia 6.4. *O parte din elementele componentei publice sunt dependente unele de altele. De aceea – fără a genera nici o ambiguitate – se poate defini componenta publică a unei scheme Boneh - Franklin sub forma*

$$BF_{public} = (G_1, G_T, \hat{e}, n, Q, H_1, H_2)$$

[1] De fapt există două scheme Boneh Franklin, apărute în [8] respectiv [9]. Deşi aparent a doua (numită în literatură $BB1$) pare mai sigură, ulterior Klitz ([46]) arată că ambele sunt echivalente din punct de vedere al securităţii.

Descrierea schemei BF

Odată fixate componentele public/secret ale schemei, utilizatorul *Bob* – având identitatea ID (scrisă ca o secvenţă binară) va calcula

$$Q_{ID} = H_1(ID)$$

şi va determina (singur sau cu ajutorul PKG) cheia secretă sQ_{ID} (care este un punct pe curba E).

Fie mesajul $M \in \{0,1\}^*$ care trebuie transmis de *Alice* lui *Bob*.
Criptarea lui se face după următorul algoritm:

Algoritmul de criptare BF **(schema simplă)**

1. Se generează aleator $r \in Z_p^*$ şi se calculează rP;
2. Se calculează Q_{ID}, după care se află
$$K = H_2(\hat{e}(rQ_{ID}, Q))$$
3. Mesajul criptat este
$$C = (C_1, C_2)$$
unde $C_1 = rP,\ C_2 = M \oplus K$.

Bob primeşte mesajul $C = (C_1, C_2) = (rP, M \oplus H_2(\hat{e}(rQ_{ID}, Q))$. El va decripta după următorul algoritm:

Algoritmul de decriptare BF **(schema simplă)**

1. Se calculează $K = H_2(\hat{e}(sQ_{ID}, C_1))$ din componenta C_1 şi cheia secretă sQ_{ID};
2. Se află $M = C_2 \oplus K$.

Schema este consistentă, deoarece $Q = sP$ şi *Alice* calculează

$$K = H_2(\hat{e}(rQ_{ID}, Q) = H_2(\hat{e}(Q_{ID}, P)^{rs})$$

iar *Bob* determină

$$K = H_2(\hat{e}(sQ_{ID}, C_1)) = H_2(\hat{e}(Q_{ID}, P)^{rs})$$

Exemplul 6.3. *Fie q un număr prim, $q \equiv 11\ (mod\ 12)$ şi curba eliptică*

$$E: \quad y^2 \equiv x^3 + 1 \pmod{q}$$

iar G_1 un subgrup de ordin P al lui $E(Z_q)$.

Folosind o funcţie de dispersie criptografică H, construim o funcţie de dispersie

$$H_1 : \{0,1\}^* \longrightarrow G_1$$

astfel:

1. *Considerăm $H(ID)$ ca un număr întreg şi calculăm restul său modulo q. Acesta este coordonata y a unui punct $Q_1 \in E(Z_q)$.*

2. *Abscisa lui Q_1 se determină cu relaţia*

$$x \equiv (y^2 - 1)^{1/3} \pmod q$$

în felul următor: din relaţia lui Euler $q^{q-1} \equiv 1 \pmod q$ rezultă $a^{2q-1} \equiv a \pmod q$. Deci, dacă $3 \mid (2q-1)$, atunci

$$a^{(2q-1)/3} \equiv a^{1/3} \pmod q$$

Cum $q \equiv 11 \pmod{12}$, acest lucru este posibil, deci

$$x \equiv (y^2 - 1)^{(2q-1)/3} \pmod q$$

3. *În final*

$$Q_{ID} = H_1(ID) = \frac{card(E(Z_q))}{p} Q_1$$

Cum $q \equiv 11 \pmod{12}$, curba E are $q+1$ elemente; deci $Q_{ID} \in E(Z_q)[p]$ este dat de relaţia

$$Q_{ID} = \frac{q+1}{p} Q_1$$

Din condiţiile $p | card(E(Z_q))$ şi $p^2 \nmid card(E(Z_q))$ rezultă că există un subgrup unic de ordin p în $E(Z_q)$, deci $Q_{ID} \in G_1$ va fi unic determinat.

Observaţia 6.5. *Să presupunem că vrem să simplificăm algoritmul prin eliminarea utilizării funcţiei de dispersie H_1, şi să folosim doar un standard H care aplică ID într-un număr întreg t, după care se consideră punctul tP drept cheie publică. Aceasta va permite unui adversar să calculeze secretul $\hat{e}(P,P)^{rst}$ cu ajutorul relaţiei*

$$(\hat{e}(rP, sP))^t = \hat{e}(P,P)^{rst}$$

deci o falie în securitatea schemei Boneh - Franklin.

Exemplul 6.4. *Elementele grupului G_T sunt din corpul finit $GF(q^k)$, deci ele se pot reprezenta sub forma*

$$\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_k), \qquad \alpha_i \in Z_q, \ \ (1 \leq i \leq k)$$

Pentru un text clar $M \in \{0,1\}^*$, o modalitate de a construi o funcţie de dispersie $H_2 : \{0,1\}^n \longrightarrow G_T$ poate consta în concatenarea coordonatelor lui α, aplicarea unei funcţii de dispersie standard $H(\alpha_1 \parallel \alpha_2 \parallel \ldots \parallel \alpha_k)$ şi reducerea ei la intervalul de valori $[0, 2^n - 1]$ – eventual prin trunchiere la primii n biţi.

Exemplul 6.5. *Să presupunem că Alice vrea să folosească schema BF pentru a cripta un mesaj folosind iniţializările din Exemplul 6.3, cu $q = 131$, $p = 11$, $P = (98, 58) \in E(Z_{131})[11]$, $G_1 =< P >$ şi $G_T =< \hat{e}(P,P) >$, unde $\hat{e} : G_1 \times G_1 \longrightarrow G_T$ este definită folosind o funcţie de distorsiune (pentru detalii, a se vedea Anexa 1)*

$$\hat{e}(P,Q) = e(P, \phi(Q))^{1560}, \quad \phi(x,y) = (\xi x, y), \ \ \xi = 65 + 112i$$

Fie cheia secretă $s = 7$; deci $Q = sP = (33, 100)$.
Să presupunem de asemenea că din identitatea lui Bob se obţine informaţia $H_2(ID_{Bob}) = Q_{ID} = (128, 57)$, deci cheia privată a lui Bob este $sQ_{ID} = (113, 8)$.

Alice poate folosi informaţiile publice pentru a cripta mesajul M (pentru Bob) astfel:

Fie $r = 5 \in Z_{11}^*$ valoarea generată aleator, pe baza căreia Alice calculează:
$rQ_{ID} = 5 \cdot (128, 57) = (98, 73)$,
$rP = 5 \cdot (98, 58) = (34, 23)$,
$K = H_2(\hat{e}(rQ_{ID}, Q) = H_2(\hat{e}(98, 73), (33, 100))) = H_2(49 + 58i)$.

Cu ajutorul acestor date, Alice determină textul criptat $C = (C_1, C_2)$ unde $C_1 = rP$, $C_2 = M \oplus K$.

La primirea mesajului criptat C, Bob calculează întâi

$$K = H_2(\hat{e}(sQ_{ID}, C_1)) = H_2(\hat{e}(113, 8), (34, 23))) = H_2(49 + 58i),$$

cu ajutorul căruia găseşte textul clar M prin

$$M = C_2 \oplus K = (M \oplus K) \oplus K = M$$

Observaţia 6.6. *Este interesant de comparat această schemă Boneh-Franklin cu algoritmul de criptare El Gamal ([2], pag. 163-175). O variantă a sistemului El Gamal este:*

Fie p un număr prim, $\alpha \in Z_p^*$ şi $\beta = \alpha^a$ elemente publice, iar $a \in Z_{p-1}$ cheia secretă. Criptarea unui text clar x de n biţi se face prin perechea $y = (y_1, y_2)$, definită

$$y_1 = \alpha^t, \qquad y_2 = x \oplus H(\beta^t)$$

unde t este un parametru aleator ales de Alice, iar $H : Z_p^* \longrightarrow \{0,1\}^n$ este o funcţie de dispersie.

Bob efectuează decriptarea după formula

$$x' = y_2 \oplus H(y_1^a)$$

Schema Boneh - Franklin de bază operează similar, înlocuind pe β cu valoarea unei perechi biliniare, calculată folosind ID şi α.

6.3.2 Schema Boneh - Franklin generalizată

Deoarece cheia K nu depinde de textul clar M, varianta simplă a schemei BF este vulnerabilă la un atac cu text clar ales.

Astfel, să presupunem că *Oscar* interceptează mesajul $C = (C_1, C_2)$, care criptează textul clar M. El va alege aleator o valoare β, va solicita decriptarea mesajului $(C_1, C_2 \oplus \beta)$ şi va primi $M \oplus \beta$ din care va afla imediat textul clar $M = (M \oplus \beta) \oplus \beta$.

Această slăbiciune este eliminată de obicei folosind transformarea Fujisaki - Okamoto ([34]).

Transformarea Fujisaki - Okamoto

Această operaţie transformă un algoritm de criptare cu cheie publică având unele falii de securitate, într-un algoritm rezistent la atacurile cu text clar ales.

Fie $E(P, X, r)$ un algoritm de criptare cu cheie publică care criptează textul clar X folosind cheia publică P şi o valoare aleatoare r, şi D algoritmul de decriptare asociat: $D(E(P, X, r)) = X$.

Se mai definesc două funcţii de dispersie H_1 şi H_2.

Atunci, pentru un text clar M, un algoritm de criptare E^* rezistent la atacul cu text clar ales poate fi definit prin

$$E^*(P, M, r) = (C_1, C_2)$$

unde $C_1 = E(P, r, H_1(r, M))$, $C_2 = H_2(r) \oplus M$.

Decriptarea unui mesaj criptat cu această schemă se realizează folosind următorul algoritm:

1. Se determină $D(C_1) = s$;
2. Se calculează $H_2(s) \oplus C_2 = M$;
3. Se determină $r = H_1(s, M)$;
4. **Dacă** $E(P, s, r) = C_1$ **atunci** exit M
   altfel Eroare, Stop.
5. Stop.

Transformarea Fujisaki - Okamoto se poate aplica direct schemei BF simple, sau se poate construi o schemă Boneh - Franklin mai generală, rezistentă la atacul cu text clar ales, care să utilizeze ideile din această transformare.

Alegerea parametrilor de securitate pentru schema BF generalizată

Fie

- O curbă eliptică E peste Z_q, (q număr prim);
- n - număr întreg;
- p – număr prim astfel ca $p|card(E(Z_q))$, $p^2 \not| card(E(Z_q))$
- Un punct $P \in E(Z_q)[p]$ ales aleator; se generează $G_1 =< P >$;
- Fie k gradul de acoperire al curbei E; se alege o pereche biliniară Tate $\hat{e} : G_1 \times G_1 \longrightarrow GF(g^k)^*$;
- Se generează $G_T =< \hat{e}(P, P) >$;
- $s \in Z_p^*$ generat aleator; se calculează $Q = sP$;
- Patru funcţii de dispersie

$$H_1 : \{0,1\}^* \longrightarrow G_1, \qquad H_2 : G_T \longrightarrow \{0,1\}^*,$$
$$H_3 : \{0,1\}^n \times \{0,1\}^n \longrightarrow Z_p^*, \quad H_4 : \{0,1\}^n \times \{0,1\}^n \longrightarrow Z_p^*$$

Componenta publică este

$$BF_{public} = (G_1, G_T, \hat{e}, n, P, Q, H_1, H_2, H_3, H_4)$$

iar cea secretă: s.

Observaţia 6.7. *În general se solicită ca numerele prime p şi q să îndeplinească anumite condiţii de securitate. Pentru aceasta, autorii recomandă ca cei doi parametri să fie numere prime Solinas.*

Definiţia 6.1. *Un număr prim p este număr prim Solinas dacă poate fi pus sub forma*

$$p = 2^a \pm 2^b \pm 1, \quad a, b \in \mathcal{N}$$

Exemplul 6.6. *Numere prime Solinas:*

$$41 = 2^5 + 2^3 + 1, \qquad 29 = 2^5 - 2^2 + 1.$$

Astfel de numere prime sunt utilizate frecvent în algoritmi IBE, deoarece minimizează numărul de calcule (sunt necesare numai adunări şi înmulţiri cu puteri ale lui 2).

Descrierea schemei Boneh - Franklin generalizate

Calculul cheii secrete se realizează identic cu varianta simplă a schemei BF: userul Bob cu identitatea ID calculează

$$Q_{ID} = H_1(ID)$$

şi apoi determină cheia secretă sQ_{ID} (punct pe curba eliptică E).

Fie mesajul $M \in \{0,1\}^n$ care trebuie transmis de *Alice* lui *Bob*.
Criptarea lui se face după următorul algoritm:

Algoritmul de criptare BF (generalizat)

1. Se calculează $Q_{ID} = H_1(ID)$;
2. Se generează aleator $\beta \in \{0,1\}^n$;
3. Se determină $r = H_3(\beta, M)$;
4. Se determină $C_1 = rP$;
5. Se determină $C_2 = \beta \oplus H_2(\hat{e}(rQ_{ID}, Q))$;
6. Se determină $C_3 = M \oplus H_4(\beta)$;
7. Textul criptat este $C = (C_1, C_2, C_3)$.

Pentru decriptarea mesajului $C = (C_1, C_2, C_3)$ *Bob* va aplica următorul algoritm:

Algoritmul de decriptare BF (generalizat)

1. Se calculează $\beta = C_2 \oplus H_2(\hat{e}(sQ_{ID}, C_1))$;
2. Se determină textul clar $M = C_3 \oplus H_4(\beta)$;
3. Se calculează $r = H_3(\beta, M)$ şi apoi se determină valoarea rP;
4. **Dacă** $C_1 = rP$ **atunci** exit M
   altfel Eroare, Stop.
5. Stop.

Securitatea schemei BF

Deoarece putem scrie $Q_{ID} = tP$ pentru o valoare oarecare (necunoscută) t, vom avea

$$\hat{e}(rQ_{ID}, Q) = \hat{e}(rtP, sP) = \hat{e}(P, P)^{rst}$$

Deci putem considera textul criptat ca fiind $C = (rP, M \oplus H_2(\hat{e}(P,P)^{rst})$.

Oscar poate obţine P şi Q din informaţia publică, poate calcula $Q_{ID} = tP$ şi mai are – din textul criptat – pe rP. Dacă din aceste elemente poate deduce $\hat{e}(P,P)^{rst}$, atunci va deduce textul clar folosind relaţia

$$(M \oplus H_2(\hat{e}(P,P)^{rst}) \oplus H_2(\hat{e}(P,P)^{rst}) = M$$

Dar calculul valorii $\hat{e}(P,P)^{rst}$ în acest mod este echivalent cu $BDHP$ (Problema Diffie - Hellman biliniară).

Deci, dacă $BDHP$ este suficient de grea, atunci va fi dificil pentru un adversar să găsească textul clar plecând de la textul criptat corespunzător.

6.4 Schema Sakai - Kasahara

Schema Sakai - Kasahara este în general o schemă rapidă de criptare. Algoritmul de calcul al cheilor este inspirat de lucrările lui Sakai şi Kasahara; de fapt schema descrisă în [17] este diferită de cea propusă de titulari.

Ca şi la schema Boneh - Franklin, sunt de fapt două versiuni ale algoritmului: una simplă (de bază), utilizată în multe standarde criptografice, şi o schemă completă, care asigură o securitate sporită.

6.4.1 Schema Sakai - Kasahara, varianta de bază

Această variantă foloseşte un secret partajat între *Alice* şi *Bob*: expeditorul *Alice* calculează componenta sa secretă folosind parametrii publici şi identitatea ID a lui *Bob*, iar destinatarul *Bob* calculează partea sa de secret, rezultată din cheia sa privată şi textul criptat.

Alegerea parametrilor de securitate pentru schema SK

Fie[2]

- Un număr întreg n (lungimea textului clar) şi un număr q – putere a unui număr prim;
- O curbă eliptică E peste un corp Z_q, având gradul de acoperire k;
- p – număr prim, cu $p|\ card(E(Z_q))$ şi care îndeplineşte anumite condiţii de securitate (de obicei se recomandă ca p să fie număr prim Solinas);

[2]Deoarece s-a folosit ca bază de lucru o curbă eliptică (structurată cu un grup aditiv), schema SK va fi reprezentată sub o formă aditivă. Unele referinţe, precum şi varianta generalizată, utilizează ca reprezentare forma multiplicativă.

- $P \in E(Z_q)[p]$ ales aleator şi $G_1 =< P >$;
- O pereche biliniară $\hat{e} : G_1 \times G_1 \longrightarrow GF(q^k)^*$;
- $v = \hat{e}(P, P)$ şi $G_T =< v >$;
- $s \in Z_p^*$ generat aleator; se calculează $Q = sP$;
- Funcţiile de dispersie

$$H_1 : \{0,1\}^* \longrightarrow Z_p, \qquad H_2 : G_T \longrightarrow \{0,1\}^n$$

Parametrii publici sunt

$$SK_{public} = (G_1, G_T, \hat{e}, n, P, Q, H_1, H_2, H_3, H_4)$$

(celelalte informaţii: p, q, v se deduc din acestea) iar componenta secretă este s.

Descrierea schemei SK

După stabilirea parametrilor şi listarea componentei publice, *Bob* îşi face cunoscută identitatea ID (public) şi îsi determină cheia privată după formula

$$D_{ID} = \frac{1}{s + H_1(ID)} \cdot P$$

Fie $M \in \{0,1\}^n$ mesajul pe care *Alice* vrea să-l trimită destinatarului cu identitatea ID. Ea va folosi următorul algoritm:

Algoritmul de criptare SK **(bază)**

1. Se calculează $q_{ID} = H_1(ID)$;
2. Se generează aleator $r \in Z_p$;
3. Se calculează $U = r \cdot (Q + q_{ID}P)$
4. Se calculează $K = H_2(v^r)$;
5. Se determină $V = M \oplus K$;
6. Textul criptat este $C = (U, V)$.

La primirea mesajului criptat $C = (U, V)$, destinatarul *Bob* va urma algoritmul:

Algoritmul de decriptare SK (bază)

1. Se calculează $K = H_2(\hat{e}(U, D_{ID}))$;
2. Se determină $M = V \oplus K$.

Schema este consistentă, deoarece

$$\hat{e}(U, D_{ID}) = \hat{e}\left(r(sP + q_{ID}P), \frac{1}{s + q_{ID}}P\right) = \hat{e}\left(r(s + q_{ID})P, \frac{1}{s + q_{ID}}P\right) = \hat{e}(P, P)^r = v^r$$

Exemplul 6.7. *Fie curba*

$$E: \ y^2 \equiv x^3 + 1 \pmod{131}$$

şi G_1 un subgrup de ordin 11 al lui $E(Z_{131})$, generat de punctul $P = (98, 58)$. Considerăm G_T subgrup al lui $GF(131^2)$ generat de $v = \hat{e}(P, P) = 28+93i$ unde $GF(131^2)$ este reprezentat prin $Z_{131}[i]$ $(i^2 = -1 \equiv 130 \pmod{131})$.

Se foloseşte perechea bilinară tate $\hat{e} : G_1 \times G_1 \longrightarrow G_T$ definită prin

$$\hat{e}(P, Q) = e(P, \phi(Q))^{1560}$$

unde ϕ este aplicaţia de distorsiune

$$\phi(x, y) = (\xi x, y) \text{ cu } \xi = 65 + 112i.$$

Fie $s = 7$ componenta secretă a cheii; se determină $Q = sP = (33, 100)$.

Să presupunem că pentru identitatea lui Bob avem $q_{ID} = 6$.
Cheia privată va fi atunci

$$D_{ID} = \frac{1}{s + q_{ID}}P = \frac{1}{13}P \equiv \frac{1}{2}P \pmod{11} = 6P = (34, 108)$$

Alice vrea să-i trimită lui Bob un mesaj; pentru aceasta, alege întîi $r = 5$, după care calculează

$$U = r(Q + q_{ID}P) = 5 \cdot ((33, 100) + 6(34, 108)) = (98, 73)$$

şi $v^5 = (28 + 93i)^5 = 39 + 24i$.
Determină apoi $K = H_2(39 + 24i)$, pe care îl foloseşte la determinarea celei de-a doua componente a textului criptat:

$$V = M \oplus K = M \oplus H_2(39 + 24i)$$

Alice trimite mesajul $C = (U, V)$ lui Bob.

La recepţie, Bob calculează

$$\hat{e}(U, D_{ID}) = \hat{e}((98, 73), (34, 108)) = 39 + 24i$$

din care află $K = H_2(39 + 24i)$. Mesajul clar îl obţine prin

$$V \oplus K = (M \oplus K) \oplus K = M$$

Observaţia 6.8. *Dacă vrem să folosim o pereche biliniară mai generală*

$$\hat{e} : G_1 \times G_2 \longrightarrow G_T,$$

atunci va fi necesară definirea unui parametru suplimentar $R \in E(Z_q)[p]$ *şi* $G_2 =< R >$. *După aceea se determină* $v = \hat{e}(P, R)$ *şi* $D_{ID} = \dfrac{1}{s + q_{ID}} R$.

6.4.2 Schema Sakai - Kasahara, forma generalizată

Schema de bază SK este vulnerabilă la un atac cu text criptat ales: astfel, dacă $Oscar$ vrea să decripteze textul $C = (U, V)$ care corespunde mesajului clar M, el va solicita decriptarea textului $C_1 = (U, V \oplus \alpha)$. Va primi mesajul $M_1 = M \oplus \alpha$, după care va găsi $M = M_1 \oplus \alpha$.
Această slăbiciune este eliminată prin utilizarea transformării Fujusaki-Okamoto. Se obţine astfel o variantă generalizată a schemei SK – rezistentă la atacul cu text criptat ales – pe care o vom prezenta în continuare. Deoarece descrierea sa în literatură este sub formă multiplicativă, vom folosi în continuare această reprezentare.

Alegerea parametrilor de securitate

Fie

- Un număr întreg n (lungimea textului clar) şi un număr q – putere a unui număr prim;
- O curbă eliptică E peste un corp Z_q, având gradul de acoperire k;
- p – număr prim, cu $p|\ card(E(Z_q))$ şi care îndeplineşte anumite condiţii de securitate (de obicei se recomandă ca p să fie număr prim Solinas);
- Un punct $g \in E(Z_q)[p]$ ales aleator şi $G_1 =< P >$ un grup ciclic;
- O aplicaţie biliniară $\hat{e} : G_1 \times G_1 \longrightarrow GF(q^k)^*$;
- $v = \hat{e}(g, g)$ şi $G_T =< v >$;
- $s \in Z_p^*$ generat aleator; se calculează $h = g^s \in G_1$;
- Funcţiile de dispersie

$$H_1 : \{0,1\}^* \longrightarrow Z_p, \qquad H_3 : \{0,1\}^n \longrightarrow Z_p^*,$$
$$H_2 : G_T \longrightarrow \{0,1\}^n \qquad H_4 : \{0,1\}^n \longrightarrow \{0,1\}^n$$

Parametrii publici sunt

$$SK_{public} = (G_1, G_T, \hat{e}, n, g, h, H_1, H_2, H_3, H_4)$$

iar componenta secretă este s.

Descrierea schemei Sakai - Kasahara (forma generalizată)

Extragerea cheii private este identică cu cea din schema de bază (reamintim, în această secţiune s-a trecut la notaţia multiplicativă):

$$d_{ID} = g^{\frac{1}{s+H_1(ID)}}$$

Fie $M \in \{0,1\}^n$ mesajul pe care *Alice* vrea să-l trimită destinatarului cu identitatea ID. Ea va folosi următorul algoritm:

Algoritmul de criptare SK (forma generalizată)

1. Se calculează $q_{ID} = H_1(ID)$;
2. Se generează aleator $\sigma \in \{0,1\}^n$;
3. Se calculează $r = H_3(\sigma, M)$;
4. Se calculează $U = (h \cdot g^{q_{ID}})^r$;
5. Se determină $V = \sigma \oplus H_2(v^r)$;
6. Se calculează $W = M \oplus H_4(\sigma)$;
7. Textul criptat este $C = (U, V, W)$.

La primirea mesajului criptat $C = (U, V)$, destinatarul *Bob* va urma algoritmul:

Algoritmul de decriptare SK (forma generalizată)

1. Se calculează $q_{ID} = H_1(ID)$;
2. Se calculează $\alpha = \hat{e}(U, d_{ID}))$;
3. Se calculează $\beta = V \oplus H_2(\alpha)$;
4. Se determină $M = W \oplus H_4(\beta)$.
5. **Dacă** $U = (g^s \cdot g^{q_{ID}})^r$ **atunci** exit M
   altfel Eroare, Stop.

Schema este consistentă, deoarece

- La Pasul 2, *Bob* calculează $\alpha = \hat{e}(U, d_{ID}) = \hat{e}\left(g^{r(s+q_{ID})}, g^{1/(s+q_{ID})}\right) = (\hat{e}(g,g))^r = v^r$.
- La Pasul 3: $\beta = V \oplus H_2(\alpha) = \sigma \oplus H_2(v^r) \oplus H_2(v^r) = \sigma$.
- La Pasul 4: $W \oplus H_4(\beta) = M \oplus H_4(\sigma) \oplus H_4(\sigma) = M$.

6.5 Exerciţii

6.1. *Să se calculeze simbolurile Jacobi* $\left(\frac{500}{1001}\right)$, $\left(\frac{396}{2445}\right)$, $\left(\frac{13}{10028}\right)$ *şi* $\left(\frac{998}{999}\right)$.

6.2. *Câte numere* $t \in (1, 77)$ *verifică simbolul Jacobi din Exemplul 6.1 ?*
Pentru fiecare valoare t_1 *găsită, să se determine* s_1 *şi apoi să se verifice relaţia*

$$\left(\frac{s_1 + 50}{77}\right) = +1.$$

6.3. *Folosind parametrii de securitate* $p = 19$, $q = 31$, $a = 70$, *să se genereze mesaje criptate – bazate pe schema Cocks – pentru biţii* 0 *respectiv* 1.

6.4. *Să se demonstreze proprietăţile* (A) *– referitoare la simbolurile Jacobi.*

6.5. *Arătaţi că schema* SK, *varianta de bază, este rezistentă la un atac cu text clar ales, iar schema* SK, *varianta generală, este rezistentă la un atac cu text criptat ales.*

Capitolul 7

Sisteme electronice de plată

7.1 Proprietăţi ale sistemelor electronice de plată

Sistemele electronice de plată realizează tranzacţii financiare prin intermediul informaţiei transmise prin canale de telecomunicaţie. Componentă principală a domeniului numit "*Comerţ electronic*" (Electronic Commerce), sistemele de plată electronice există în mai multe forme, printre care: *cecuri digitale* (digital checks), *cărţi de credit* (credit cards), *bani electronici* (electronic cash).

Subiectul acestui capitol este îndreptat asupra sistemelor de plată care utilizează monedele electronice şi sunt definite prin anumite proprietăţi criptografice. Utilizând acest tip de sisteme, vom încerca realizarea unui model care va imita sistemul bancar clasic de emitere a numerarului, cu singura diferenţă că totul este realizat digital.
În plus, sistemele de plată prezentate sunt de tip *off - line*.

T. Okamoto şi K. Ohta au definit în 1991 ([65]) şase proprietăţi fundamentale ale unui sistem electronic de plată:

1. **Securitate**: Protocolul de tranzacţie trebuie să asigure un nivel de securitate ridicat, utilizând tehnici criptografice. Vor fi prevenite atacuri care încearcă să modifice sau să reproducă informaţia transmisă.

2. **Anonimitate**: Permite utilizatorului să nu fie identificat în cursul desfăşurării protocolului de tranzacţie.

3. **Portabilitate**: Securitatea şi folosirea sistemului digital nu trebuie să fie dependente de caracteristicile fizice ale calculatoarelor folosite.
   În plus, informaţia va putea fi transferată prin orice reţea de calculatoare, fără să existe o reţea separată care să beneficieze de drepturi speciale.

4. **Transfer în ambele sensuri**: Monedele digitale se pot transfera direct între utilizatori, fără ca acest transfer să fie realizat printr-o bancă.

5. **Divizibilitate**: O monedă electronică obţinută de utilizator din contul său poate fi împărţită în monede de valori mai mici, cu scopul de a fi folosite (separat sau nu) la tranzacţii diferite.

6. **Capacitate de a realiza tranzacţii "off-line"**: Protocolul de tranzacţie între două părţi trebuie să se desfăşoare în orice moment, fără autentificarea unei a treia părţi şi fără ca utilizatorii să fie conectaţi obligatoriu la o instituţie financiară de control.

Scenariul unui sistem electronic de plată presupune 3 tipuri de participanţi:

- Un utilizator (sau client) pe care-l vom nota de obicei $\mathcal{U}$ (User).
- Un comerciant (sau magazin) care va accepta banii electronici ai clientului, furnizând în schimb anumite bunuri. Acesta se va nota $\mathcal{S}$ (Seller).
- O instituţie financiară – notată $\mathcal{B}$ (Bank) – unde atât clientul, cât şi comerciantul au conturi deschise. În realitate este posibil să existe o reţea de bănci care comunică şi realizează tranzacţii între ele, dar noi vom considera (pentru simplificare) existenţa unei singure bănci.

7.1.1 Securitatea plăţilor electronice

Pentru a asigura securitatea informatică a unui sistem electronic de plată, trebuie îndeplinite în primul rând condiţiile specifice sistemelor de criptare:

- **Intimitate** (Privacy) sau protecţia împotriva interceptării mesajelor.
- **Autentificarea utilizatorului**: fiecare dintre cele două părţi implicate într-un protocol trebuie să fie sigură de identitatea celeilalte părţi.
- **Integritatea mesajului**: destinatarul trebuie să verifice dacă mesajul primit a fost (sau nu) modificat în tranzit.
- **Imposibilitatea negării** (Non-repudiation): cel care a trimis un mesaj nu va putea să nege mai târziu acest lucru.

David Chaum, iniţiatorul sistemelor electronice de plată, defineşte încă două trăsături de bază ale acestora:

- Imposibilitatea băncii de a urmări utilizatorul pe baza protocolului de extragere a monedelor electronice – în cazul în care acesta este onest (adică va utiliza monedele o singură dată).

Vom vedea ulterior că pentru un sistem de încredere se impune ca banca să aibă posibilitatea de a-l identifica pe cel care comite o fraudă (încearcă să folosească aceeaşi monedă de mai multe ori). Mai mult decât atât, s-au elaborat sisteme (toate folosind o componentă hardware de tip *card, chip* sau *micro-procesor*) care previn frauda.

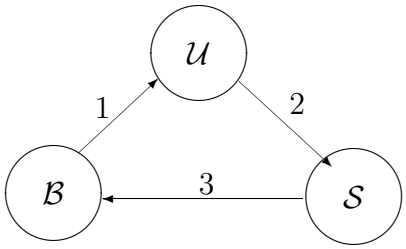
Dacă se respectă această condiţie, sistemul se va numi **fără urmărire** (*un-traceable*).

- Imposibilitatea de a deduce identitatea utilizatorului, dacă se urmăresc mai multe monede folosite de acesta; altfel spus – nu se poate realiza o legătură între monedele folosite în tranzacţii diferite pentru a se obţine informaţii suplimentare referitoare la utilizator.

  În acest caz, sistemul se va numi **fără legături** (*un-linkable*).

De remarcat că sistemul de plată folosind cărţi de credit nu dispune de aceste ultime două proprietăţi; de aceea David Chaum împreună cu alţi cercetători au propus sistemul de plată cu monede electronice, care va oferi utilizatorilor lor aceste două garanţii suplimentare.

Într-un protocol de tranzacţie cu un astfel de sistem – având ca protagonişti utilizatorul $\mathcal{U}$, banca $\mathcal{B}$ şi comerciantul/magazinul $\mathcal{S}$ – succesiunea evenimentelor este următoa-rea:



1. Protocolul de extragere al monedelor (*withdrawal protocol*):
   Utilizatorul va extrage monede de o anumită valoare din contul său din bancă şi le va stoca într-o componentă hard (harddisk sau card).

2. Protocolul de plată (*payment protocol*):
   Utilizatorul va transfera bani digitali comerciantului (sau magazinului), primind în schimb mărfuri/servicii.

3. Protocolul de depozit (*deposit protocol*):
   Comerciantul va transfera banii primiţi în contul său din bancă.

Aceste protocoale se pot implementa în două moduri:

- **on-line**: Comerciantul ia legătura cu banca şi verifică validitatea monedei trimisă de client, înainte de a-i livra acestuia bunurile.

  Acesta este modul care se utilizează în prezent de către tranzacţiile cu carduri de credit/debit.

- **off-line**: Comerciantul acceptă monedele clientului după un protocol de verificare cu acesta, fără a face apel la bancă. Protocolul de depozit la bancă se va desfăşura periodic (de exemplu la sfârşitul săptămânii).

La sistemele de tip "on-line", protocoalele de plată şi de depozit nu sunt practic separate. Cel mai mare dezavantaj al acestor sisteme – şi unul din motivele pentru care nu se folosesc la operaţiile cu sume mici – este faptul că la fiecare tranzacţie trebuie apelată banca. Cum numărul utilizatorilor de Internet şi de sisteme electronice comerciale creşte accelerat, aceasta devine o problemă dificilă de comunicare.

Tipul de fraudă cel mai frecvent întâlnit constă în utilizarea unei monede de mai multe ori (*double-spending* sau *multiple-spending*). Pentru a se proteja împotriva acestei fraude, banca va menţine o bază de date cu toate monedele electronice folosite. Executând protocolul de depozit, banca va căuta moneda primită în baza de date şi – dacă se află deja acolo – o va respinge. Dacă plata se face "on-line", în acest fel se poate preveni utilizarea aceleiaşi monede de mai multe ori. Dacă se foloseşte sistemul "off-line", se defineşte un mecanism suplimentar prin care banca va putea identifica utilizatorul care a comis frauda. Dezavantajul la sistemele "off-line" este intervalul de timp după care se face identificarea, timp în care utilizatorul necinstit poate să dispară.
De aceea sistemele "off-line" nu sunt recomandate pentru sume mari de bani.

Pentru realizarea sistemelor informatice care dispun de proprietăţile enunţate mai sus, se foloseşte ca bază de lucru criptografia cu chei publice.

7.2 Scheme de identificare

O schemă de identificare este un protocol între două părţi: *Alice* (*prover*) şi *Bob* (*verifier*) în care *Alice* demonstrează cunoaşterea cheii sale secrete fără a o releva, iar *Bob* – la sfârşitul protocolului – este convins de identitatea lui *Alice*.

Orice schemă de identificare se poate transforma într-un protocol de semnătură prin utilizarea unei funcţii de dispersie.
Vom prezenta schema de identificare şi protocolul de semnătură definite de Clauss Schnorr, a căror securitate este bazată pe problema logaritmului discret.

În protocoalele lui Schnorr se aleg:

- două numere prime p şi q, astfel încât $q|p-1$;

- un parametru de securitate t;

- un generator $g \in Z_p^*$ de ordin q ($g^q \equiv 1 \ (mod\ p)$).

Pentru securitate se recomandă ca q să aibă în reprezentarea binară minim 150 biţi, p – minim 512 biţi, iar parametrul de securitate t – în jur de 72 biţi.

Numerele (p, q, g) vor fi făcute publice.

Pentru a genera o pereche *(cheie publică, cheie privată)*, utilizatorul *Alice* alege aleator un număr $s \in Z_q^*$ şi calculează $v = g^{-s} \ (mod\ p)$.
s va fi cheia sa privată, iar v cheia sa publică.

Protocolul de autentificare:

Presupunând că *Alice* vrea să îi demonstreze identitatea sa lui *Bob*, cei doi parcurg următorul protocol:

Protocolul de autentificare Schnorr:

1. *Alice* generează aleator un număr $r \in Z_q^*$, calculează $b = g^r \ \ (mod\ p)$, pe care-l trimite lui *Bob*.
2. *Bob* trimite lui *Alice* un număr aleator $x,\ x \in (0, 2^t)$, numit *provocare*.
3. *Alice* calculează
$$y = r + sx \quad (mod\ q)$$
şi îi răspunde lui *Bob* cu y.
4. *Bob* verifică relaţia
$$b = g^y v^x \quad (mod\ p)$$
şi – dacă ea este îndeplinită – acceptă autentificarea lui *Alice*.

Protocolul se numeşte de tip **provocare/răspuns**.

Dacă *Alice* urmează acest protocol pas cu pas, atunci

$$g^y v^x \equiv g^{r+sx} g^{-sx} \equiv g^r \equiv b \quad (mod\ p)$$

deci relaţia de la pasul 4 este verificată şi utilizatorul *Alice* este autentificat.

Evident, dacă *Bob* doreşte să se autentifice în faţa lui *Alice*, va urma acelaşi protocol.

Ideea protocolului Schnorr constă în construcţia de către *Alice* a unei drepte $d : y = r + sx$ de pantă secretă s. *Alice* îi va demonstra lui *Bob* cunoaşterea dreptei d, răspunzând la o provocare x cu punctul y corespunzător de pe dreaptă. Verificând relaţia de la pasul 4, *Bob* este convins că *Alice* cunoaşte panta secretă a dreptei, fără a o putea calcula şi el.

De fiecare dată când iniţiază acest protocol, *Alice* trebuie să aleagă o nouă valoare pentru r. Altfel, *Bob* se va afla în posesia a două puncte ale dreptei, determinând astfel panta dreptei – care este chiar cheia secretă s a lui *Alice*.
Vom demonstra acest lucru:

Lema 7.1. *Executând de două ori protocolul de autentificare Schnorr cu acelaşi număr r ales la pasul 1, în final Bob va fi în posesia cheii secrete s a lui Alice.*

Demonstraţie. După pasul 1, *Bob* cunoaşte $b = g^r \ (mod\ p)$.

Fie $x_1, x_2 \in [1, 2^t)$ cele două provocări diferite trimise de *Bob* în cele două execuţii ale protocolului Schnorr. În final *Bob* va cunoaşte:

$$\begin{cases} y_1 = r + sx_1 \pmod q \\ y_2 = r + sx_2 \pmod q \end{cases}$$

Rezolvând acest sistem, se obţine

$y_1 - y_2 = s(x_1 - x_2)\ (mod\ q) \quad \Longrightarrow \quad s = (y_1 - y_2)(x_1 - x_2)^{-1}\ (mod\ q).$ □

Protocolul de semnătură:

Să arătăm cum poate fi transformat protocolul de autentificare Schnorr într-un protocol de semnătură digitală.

Introducem o funcţie de dispersie criptografică h.

Presupunem că *Alice* doreşte să obţină de la *Bob* o semnătură asupra unui mesaj m. Similar protocolului de autentificare Schnorr, *Bob* va avea cheia publică v şi cheia sa secretă s ($v = g^{-s}\ (mod\ p)$).

Protocolul de semnătură Schnorr

1. *Bob* generează aleator un număr $r \in Z_q^*$ şi calculează $b = g^r\ (mod\ p)$.
2. *Bob* concatenează m cu b şi formează $x = h(m\|b)$.
3. *Bob* calculează $y = r + sx\ (mod\ q)$ şi trimite lui *Alice* perechea (x, y).
4. *Alice* calculează
$$b' = g^y v^x\ (mod\ p) \quad \text{şi} \quad x' = h(m\|b')$$
Dacă $x = x'$, atunci *Alice* acceptă (x, y) ca o semnătură validă pentru m.

Observăm că pentru a transforma protocolul de autentificare Schnorr într-un protocol de semnătură este suficient să înlocuim provocarea cu dispersia asupra documentului care trebuie semnat.

Ideea poate fi folosită la toate protocoalele de autentificare de tip **provocare/răspuns**.

Formal, protocolul de semnătură Schnorr este definit astfel:

$\mathcal{P} = \mathcal{A} = Z_p,\ \mathcal{K} = \{(p, q, g, s, v, r) \mid s, r \in Z_q^*,\ v = g^{-s}\ (mod\ p),\ q|p-1\}$.
Dacă $K = (p, q, g, s, v, r) \in \mathcal{K}$, cheia (p, q, g, v) este publică, iar (s, r) este secretă.
Pentru $K = (p, q, g, s, v, r) \in \mathcal{K},\ m \in \mathcal{P},\ y \in \mathcal{A}$:

$x = h(m\|g^r), \quad sig_K(x) = r + sx \quad (mod\ q)$

$ver_K(x, y) = T \quad \Longleftrightarrow \quad x \equiv h(m\|g^y v^x)\ (mod\ p)$.

Dacă *Bob* urmează protocolul de semnătură Schnorr, atunci:

$$b' = g^y v^x \equiv g^{r+sx} g^{-sx} \equiv g^r \equiv b \ (mod\ p)$$
$$x' = h(m\|b') = h(m\|b) = x$$

deci relaţia de la pasul 4 este verificată şi *Alice* acceptă semnătura lui *Bob* drept validă.

Observăm că determinarea cheii secrete s din cheia publică v, cunoscând g, este echivalentă – din punct de vedere al complexităţii – cu rezolvarea problemei logaritmului discret.

7.3 Sistemul de plată Chaum - Fiat - Naor

Cele mai importante protocoale comerciale de plată sunt de tip off-line. Vom prezenta în acest capitol trei astfel de protocoale. Primul – sistemul Chaum - Fiat - Naor – este unul din cele mai simple astfel de scheme, principala sa proprietate fiind anonimitatea.

El permite unui utilizator $\mathcal{U}$ să creeze un cec bancar, fără ca banca $\mathcal{B}$ să poată determina autorul semnăturii (şi – deci – nici pe cel care a cheltuit suma respectivă la un magazin $\mathcal{S}$); singurul caz în care identitatea lui $\mathcal{U}$ poate fi dezvăluită este când acesta foloseşte acelaşi cec de două ori.

7.3.1 Descrierea sistemului Chaum - Fiat - Naor

Schema de plată Chaum - Fiat - Naor ([14]) are la bază protocolul de semnătură blind *RSA* (Capitolul 3, secţiunea 3.11.1).

Tehnica de bază folosită în construcţie este numită **Cut-and-Choose** şi poate fi descrisă astfel: când $\mathcal{U}$ doreşte să extragă o monedă din bancă, construieşte un mesaj format din k perechi de numere (unde k este numărul de variante). Aceste numere au proprietatea că dintr-o pereche se poate calcula identitatea lui $\mathcal{U}$ (de exemplu printr-un simplu XOR între cele două numere ale perechii), iar având la dispoziţie un singur număr din fiecare pereche nu se poate deduce nici o informaţie.
$\mathcal{U}$ concatenează cele k variante, după care obţine o semnătură *blind* a băncii asupra mesajului trimis, cu protocolul de semnătură blind *RSA*.

În momentul în care $\mathcal{U}$ doreşte să folosească moneda la magazinul $\mathcal{S}$, el primeşte o provocare de la $\mathcal{S}$ formată din k biţi generaţi aleator. Pentru fiecare bit, $\mathcal{U}$ răspunde cu primul număr al perechii corespunzătoare numărului bitului (dacă bitul este 0) sau cu al doilea număr – dacă bitul este 1. (De exemplu, dacă provocarea este 0110 . . . , $\mathcal{U}$ răspunde cu primul număr din prima pereche, al doilea număr din a doua pereche, al doilea număr din a treia pereche etc).
Când $\mathcal{S}$ depozitează moneda la bancă, îi trimite provocarea, împreună cu numerele generate de $\mathcal{U}$. $\mathcal{B}$ păstrează moneda într-o bază de date, împreună cu provocarea lui $\mathcal{S}$ şi răspunsul lui $\mathcal{U}$.

Dacă $\mathcal{U}$ încearcă refolosirea monedei, primeşte o nouă provocare aleatoare de k biţi, care diferă de prima în cel puţin un bit cu o probabilitate destul de mare: $1 - \frac{1}{2^k}$. Astfel, pentru bitul care diferă, $\mathcal{U}$ va revela ambele numere ale perechii corespunzătoare.
Acum $\mathcal{B}$ poate calcula din cele două numere identitatea lui $\mathcal{U}$.

Să formalizăm aceste idei.
În prima fază, banca face publice:

- Cheia (e, n) (păstrează secretă cheia d astfel ca $e \cdot d \equiv 1 \ (mod\ \phi(n))$);
- Un parametru de securitate k;
- Două funcţii de dispersie criptografică $h_1,\ h_2$.

$\mathcal{U}$ are un cont în banca $\mathcal{B}$, identificat cu numărul u, căruia banca îi asociază un contor v.

Protocolul de extragere

1. $\mathcal{U}$ generează aleator $3k$ numere $a_i, c_i, d_i \in Z_n^*$ $(1 \leq i \leq k)$.
   Calculează (pentru $i = 1, \ldots, k$):

   $$x_i = h_2(a_i, c_i), \qquad y_i = h_2(a_i \oplus (u \| (v+i)), d_i),$$

   şi crează k unităţi $U_i = h_1(x_i, y_i)$.

2. Ascunde aceste unităţi cu un protocol de semnătură blind RSA:

   $$B_i = r_i^e \cdot U_i \ \ (mod\ n) \qquad (1 \leq i \leq k)$$

   unde factorii $r_1, r_2, \ldots, r_k \in Z_n^*$ sunt generaţi aleator.
   Trimite băncii $\mathcal{B}$ valorile $B_1, B_2, \ldots, B_k$.

3. $\mathcal{B}$ alege aleator $\lfloor k/2 \rfloor$ valori pentru test.
   Pentru uşurinţa prezentării, presupunem că sunt primele $\lfloor k/2 \rfloor$ valori.

4. $\mathcal{U}$ trimite lui $\mathcal{B}$ valorile $r_i, a_i, c_i, d_i \quad (1 \leq i \leq k/2)$.

5. $\mathcal{B}$ dezvăluie aceste $\lfloor k/2 \rfloor$ unităţi şi verifică dacă $\mathcal{U}$ a încercat să trişeze. Dacă totul este corect, semnează celelalte $\lceil k/2 \rceil$ unităţi, trimite lui $\mathcal{U}$ valoarea

   $$D = \prod_{j=\lceil k/2 \rceil}^{k} (B_j)^d \quad (mod\ n)$$

   şi debitează contul lui $\mathcal{U}$ cu suma solicitată.

6. $\mathcal{U}$ calculează moneda

$$C = D \cdot \prod_{j=\lceil k/2 \rceil}^{k} r_j^{-1} = \prod_{j=\lceil k/2 \rceil}^{k} U_j^d \pmod n$$

şi incrementează cu k copia proprie a contorului v.

Protocolul de plată

1. $\mathcal{U}$ trimite lui $\mathcal{S}$ moneda C şi unităţile $U_1, \ldots, U_k$.
2. $\mathcal{S}$ alege o aplicaţie injectivă $\tau : [1, \lfloor k/2 \rfloor] \longrightarrow [1, k]$ şi generează aleator şirul binar $z = z_{\tau(1)}, z_{\tau(2)}, \ldots, z_{\tau(\lfloor k/2 \rfloor)}$.
   Trimite această provocare (aplicaţia τ şi şirul z) lui $\mathcal{U}$.
3. Pentru $i = 1, 2, \ldots, \lfloor k/2 \rfloor$, $\mathcal{U}$ trimite lui $\mathcal{S}$:

$$\begin{cases} (y_{\tau(i)}, a_{\tau(i)}, c_{\tau(i)}) & \text{dacă } z_{\tau(i)} = 1 \\ (x_{\tau(i)}, a_{\tau(i)} \oplus (u \| (v + \tau(i))), d_{\tau(i)}) & \text{dacă } z_{\tau(i)} = 0 \end{cases}$$

4. $\mathcal{S}$ verifică validitatea monedei C; dacă este corectă, o acceptă ca plată.

Validitatea monedei C este calculată astfel:

- Dacă $z_{\tau(i)} = 1$, atunci $\mathcal{S}$ determină $x_{\tau(i)} = h_2(a_{\tau(i)}, c_{\tau(i)})$;
- Dacă $z_{\tau(i)} = 0$, atunci $\mathcal{S}$ determină $y_{\tau(i)} = h_2(a_{\tau(i)} \oplus (u \| (v + \tau(i))), d_{\tau(i)})$;

În etapa a doua, $\mathcal{S}$ verifică egalităţile $U_{\tau(i)} = h_1(x_{\tau(i)}, y_{\tau(i)})$, $1 \leq i \leq \lfloor k/2 \rfloor$.

Protocolul de depozit

1. $\mathcal{S}$ trimite – lui $\mathcal{B}$ – moneda C împreună cu detaliile protocolului de plată.
2. $\mathcal{B}$ verifică semnătura sa pe moneda C.
3. $\mathcal{B}$ verifică dacă C nu a mai fost cheltuită anterior.
4. $\mathcal{B}$ depune C într-o bază de date cu monedele cheltuite, împreună cu datele folosite în protocolul provocare/răspuns din protocolul de plată.
5. $\mathcal{B}$ depune în contul lui $\mathcal{S}$ suma corespunzătoare monedei C.

7.3.2 Securitatea sistemului de plată Chaum - Fiat - Naor

Modelul Chaum - Fiat - Naor satisface primele patru proprietăţi definite în primul paragraf: securitate (garantată de o valoare mare a lui k), anonimitate ($\mathcal{B}$ poate afla identitatea lui $\mathcal{U}$ doar dacă acesta încearcă să cheltuie o monedă de două ori), portabilitate, transfer în ambele sensuri (o monedă este cheltuită o singură dată; după tranzacţie ea este returnată băncii şi depozitată într-o bază de date).

Puncte slabe ale schemei:

- Nu este posibilă divizarea monedei în valori mai mici.
- Fiecare monedă trebuie însoţită de $2k$ numere mari: $U_1, \ldots, U_k$, aplicaţia τ, provocările $z_{\tau(i)}$ şi răspunsurile lui $\mathcal{U}$ la aceste provocări.

Atacuri posibile:

- Orice atac asupra sistemului de criptare RSA poate fi utilizat ca un potenţial atac asupra sistemului de plată Chaum - Fiat - Naor.
- Schema nu rezistă unui *atac prin cooperare*: după tranzacţie, $\mathcal{U}$ poate trimite altui vânzător $\mathcal{S}'$ moneda folosită, împreună cu secvenţa binară generată de $\mathcal{S}$ şi răspunsul său la această provocare.
  Deşi vede că s-a depus aceeaşi monedă, $\mathcal{B}$ nu poate decide care din magazinele $\mathcal{S}, \mathcal{S}'$ a trişat – şi trebuie să le crediteze pe amândouă cu aceeaşi sumă.

7.4 Schema de plată Ferguson

Este un sistem de plată definit de Niels Ferguson în 1993 ([32]), fundamentat pe partajare de secrete. Semnătura blind folosită este numită *semnătură blind aleatoare bazată pe RSA*. În acest caz

- $\mathcal{U}$ primeşte o semnătură RSA pe un număr de o formă specială, pe care nu îl poate crea singur.
- $\mathcal{B}$ este sigur că numărul pe care îl semnează (blind) a fost generat aleator.
- $\mathcal{B}$ nu are nici o informaţie despre semnătura obţinută de $\mathcal{U}$.

Protocolul de semnătură blind aleatoare

1. $\mathcal{B}$ face publice: cheia sa (ν, n), funcţia de dispersie criptografică h, şi numărul aleator $g \in Z_n^*$.
2. $\mathcal{U}$ generează aleator $a_1 \in Z_n^*$ şi doi factori "de orbire" σ, γ.
   Trimite lui $\mathcal{B}$ produsul $\gamma^\nu a_1 g^\sigma$.
3. $\mathcal{B}$ generează aleator $a_2 \in Z_n^*$, pe care îl trimite lui $\mathcal{U}$.
4. $\mathcal{U}$ răspunde cu $h(a) - \sigma$, unde $a = a_1 \cdot a_2$.
5. $\mathcal{B}$ calculează şi trimite lui $\mathcal{U}$
$$\left[(\gamma^\nu a_1 g^\sigma) \cdot a_2 \cdot g^{h(a)-\sigma}\right]^{1/\nu} = \gamma \left(a g^{h(a)}\right)^{1/\nu}$$
6. $\mathcal{U}$ împarte acest număr la γ şi obţine mesajul semnat
$$\left(a, \left(a g^{h(a)}\right)^{1/\sigma}\right)$$

7.4.1 Structura sistemului de plată Ferguson

Sistemul de plată utilizează următoarele numere, făcute publice de banca $\mathcal{B}$:
- $g_a, g_b, g_c \in Z_n^*$ elemente de ordin mare,
- $u_b, u_c \in Z_n^*$ generatori (elemente de ordin n).

Se mai folosesc funcţiile de dispersie criptografică h_1, h_2, cheia publică (ν, n) a băncii şi ID - identitatea utilizatorului $\mathcal{U}$.

Cele trei protocoale care formează schema de plată Ferguson sunt:

- **Protocolul de extragere**: Constă în esenţă din trei parcurgeri în paralel ale schemei de semnătură blind aleatoare:

1. $\mathcal{U}$ generează aleator $a_1, b_1, c_1, \alpha, \beta, \gamma \in Z_n^*$, $\sigma, \tau, \phi \in Z_\nu^*$.
   Apoi calculează şi trimite lui $\mathcal{B}$ valorile

$$\alpha^\nu a_1 g_a^\tau, \quad \beta^\nu b_1 g_b^\phi, \quad \gamma^\nu c_1 g_c^\alpha \quad \text{(calcule modulo } n\text{)}$$

2. $\mathcal{B}$ generează $a_2, b_2, c_2 \in Z_n^*$ şi trimite lui $\mathcal{U}$ tripletul $(a_2, u_b^{b_2}, u_c^{c_2})$.
3. $\mathcal{U}$ generează $k_1 \in Z_\nu^*$ şi calculează

$$\begin{aligned} e_c &= h_1(u_c^{c_1 c_2}) - \sigma, & e_b &= h_1(u_b^{b_1 b_2}) - \phi, \\ a &= (a_1 a_2 h_2(e_c, e_b)))^{k_1}, & e_a &= \frac{h_1(a)}{k_1} - \tau \end{aligned}$$

   Valorile (e_a, e_b, e_c) sunt trimise lui $\mathcal{B}$.
4. $\mathcal{B}$ calculează

$$\begin{aligned} C' &= \gamma^\nu c_1 g_c^\sigma c_2 g_c^{e_c} = \gamma^\nu c g_c^{h_1(u_c^c)}, && \text{unde } c = c_1 c_2 \\ \beta' &= \beta^\nu b_1 g_b^\phi b_2 g_b^{e_b} = \beta^v b g_b^{h_1(u_b^b)}, && \text{unde } b = b_1 b_2 \\ A' &= \alpha^\nu a_1 g_a^\tau a_2 h_2(e_c, e_b) g_a^{e_a} = \alpha^\nu a g_a^{h_1(a)/k_1} \end{aligned}$$

   Apoi alege aleator $k_2 \in Z_\nu^*$ şi trimite lui $\mathcal{U}$ cinci valori:

$$c_2,\ b_2,\ k_2,\ (C')^{k_2/\nu} \cdot (A')^{1/\nu},\ (C')^{U/\nu} \cdot (B')^{1/\nu}.$$

5. $\mathcal{U}$ construieşte numerele

$$A = a \cdot g_a^{h_1(a)}, \qquad B = b \cdot g_b^{h_1(u_b^b)}, \qquad C = c \cdot g_c^{h_1(u_c^c)}$$

   şi apoi generează semnăturile

$$S_a = \left(\frac{(C')^{k_2/\nu} (A')^{1/\nu}}{g^{k_2} \alpha} \right)^{k_1}, \qquad S_b = \frac{(C')^{U/\nu} (B')^{1/\nu}}{\gamma^U \beta}$$

   Verifică corectitudinea lor cercetând dacă următoarele egalităţi sunt adevărate:

$$S_b^\nu = C^U B, \qquad S_a^\nu = C^k A \ \text{ unde } k = k_1 k_2$$

   În caz afirmativ, $\mathcal{U}$ termină protocolul de extragere cu numerele

$$(a, b, c), (S_a, S_b), k$$

Observaţia 7.1.

– *Între valorile A, B, C şi corespondenţele lor blind există relaţiile*

$$C' = \gamma^{\nu} C, \qquad A' = \alpha^{\nu} A^{1/k}, \qquad B' = \beta^{\nu} B.$$

– *Consistenţa relaţiilor de verificare de la pasul* 5*:*

$$S_b^{\nu} = \frac{(C')^U B'}{\gamma^{U\nu}\beta^{\nu}} = \frac{(\gamma^{\nu} C)^U \beta^{\nu} B}{\gamma^{U\nu}\beta^{\nu}} = C^U B,$$

$$S_a^{\nu} = \frac{(C')^k (A')^{k_1}}{\gamma^{k\nu}\alpha^{k_1\nu}} = \frac{\gamma^{k\nu} C^k \alpha^{k_1\nu} A}{\gamma^{k\nu}\alpha^{k_1\nu}} = C^k A$$

- **Protocolul de plată**

1. $\mathcal{U}$ trimite lui $\mathcal{S}$ moneda (a, b, c).
2. $\mathcal{S}$ lansează o provocare $x \in Z_{\nu}$, generată aleator.
3. $\mathcal{U}$ răspunde cu (r, R) unde $r = ID \cdot x + k$ şi $R = (C^r A^x B)^{1/\nu}$.
4. $\mathcal{S}$ calculează A, B, C şi verifică dacă

$$R^{\nu} = C^r A^x B.$$

În caz afirmativ, acceptă moneda şi oferă bunurile/serviciile solicitate.

- **Protocolul de depozit**

1. $\mathcal{U}$ trimite lui $\mathcal{B}$ moneda (a, b, c), provocarea x şi răspunsul (r, R).
2. $\mathcal{B}$ verifică corectitudinea monedei şi creditează $\mathcal{S}$ cu valoarea sa (valoarea monedei depinde de cheia publică ν).

7.4.2 Securitatea sistemului de plată Ferguson

Sistemul previne dubla utilizare a unei monede: dacă $\mathcal{U}$ utilizează aceeaşi monedă de două ori, banca va avea două informaţii de tipul

$$r_1 = ID \cdot x_1 + k, \qquad r_2 = ID \cdot x_2 + k$$

(în care $x_1 \neq x_2$), sistem din care va putea deduce identitatea ID a utilizatorului.

Sunt îndeplinite primele patru proprietăţi definite pentru sistemele de plată: securitate, anonimitate (ambele bazate pe semnătura blind aleatoare), portabilitate (fiecare monedă este formată din 3 numere şi două semnături şi poate fi stocată în aproximativ 250 octeţi), transferabilitate (cu remarca că transferul unei monede prin mai mulţi utilizatori duce la creşterea dimensiunii, deoarece ea trebuie să conţină informaţii despre toţi utilizatorii care au folosit-o).

Ca tipuri de atac, putem menţiona:

- Orice atac asupra unui sistem de criptare RSA poate fi remodelat ca un atac asupra sistemului de plată Ferguson.
- Sistemul este permeabil la un *atac prin cooperare*, construit identic cu cel asupra sistemului de plată Chaum - Fiat - Naor. Pentru a preveni, autorul sugerează utilizarea provocării din protocolul de plată sub forma unei dispersii a valorii monedei şi identităţii lui $\mathcal{S}$.

Deşi schema Ferguson evită tehnica Cut - and - Choose folosită de protocolul Chaum - Fiat - Naor, ea este una din cele mai complicate scheme de plată, din cauza semnăturii blind aleatoare bazată pe RSA.
Tot din cauza RSA, ea nu poate fi implementată pe curbe eliptice.

7.5 Problema reprezentării în grupuri

Vom vom defini o nouă problemă $\mathcal{NP}$ - completă – problema reprezentării în grupuri; ea va constitui baza construcţiei ultimului sistem de plată electronic prezentat – sistemul Brands.
Vom demonstra, de asemenea, echivalenţa acestei probleme – din punct de vedere al complexităţii – cu problema logaritmului discret.

7.5.1 Definirea problemei reprezentării

Definiţia 7.1. *Fie $k \geq 2$ număr natural şi q un număr prim.*
Un tuplu generator de lungime k pentru grupul Z_q este un k-tuplu $(g_1, \ldots, g_k)$ astfel încât $g_i \in Z_q \backslash \{0, 1\}$, $\forall i \in \{1, 2, \ldots, n\}$ şi $g_i \neq g_j$, $\forall i \neq j$.

Pentru orice $\alpha \in Z_q^*$, $(a_1, \ldots, a_k)$ este o reprezentare a lui α în raport cu tuplul generator $(g_1, \ldots, g_k)$ dacă

$$\alpha = \prod_{i=1}^{k} g_i^{a_i} \pmod q$$

Lema 7.2. *Fie $k \geq 2$ un număr natural şi q un număr prim. Pentru orice $\alpha \in Z_q$ şi orice $(g_1, \ldots, g_k)$ tuplu generator, există exact q^{k-1} reprezentări ale lui α în raport cu $(g_1, \ldots, g_k)$.*

Demonstraţie. Cum q este prim, orice element din Z_q este primitiv, deci generator al grupului multiplicativ Z_q^*.
Generăm aleator primele $k-1$ elemente ale unui k-tuplu $(a_1, \ldots, a_k)$ in q^{k-1} moduri; apoi, din relaţia

$$\prod_{i=1}^{k} g_i^{a_i} = \alpha \pmod q$$

determinăm

$$g_k^{a_k} = \alpha \left(\prod_{i=1}^{k-1} g_i^{a_i} \right)^{-1} \pmod q$$

deci

$$a_k = log_{g_k} \left[\alpha \left(\prod_{i=1}^{k-1} g_i^{a_i} \right)^{-1} \right] \pmod q$$

Logaritmul are sens, deoarece din definiţia tuplului generator, $g_k \neq 1$.

Din problema logaritmului discret, a_k este unic determinat, deci există exact q^{k-1} reprezentări ale lui α în raport cu $(g_1, \ldots, g_k)$. □

Acest rezultat relativ simplu arată că, fixând un număr $\alpha \in Z_q^*$ şi un tuplu generator $(g_1, \ldots, g_k)$, printr-o căutare exhaustivă în mulţimea k-tuplurilor peste Z_q, probabilitatea de a obţine o reprezentare a lui α în raport cu $(g_1, \ldots, g_k)$ este $\frac{q^{k-1}}{q^k} = \frac{1}{q}$.

Astfel, suntem în măsură să enunţăm problema reprezentării în grupuri având un număr prim de elemente.

Problema reprezentării:
Fie q un număr prim, $k \geq 2$ un număr natural, $(g_1, \ldots, g_k)$ un tuplu generator şi $\alpha \in Z_q^*$.
Să se găsească o reprezentare a lui h în raport cu $(g_1, \ldots, g_k)$ în Z_q, în cazul în care aceasta există.

Observăm că pentru $k = 1$ se obţine problema logaritmului discret; de aceea – pentru a face distincţie între cele două probleme – am considerat $k \geq 2$.

7.5.2 Echivalenţa problemei reprezentării cu problema logaritmului discret

Să reamintim câteva definiţii referitoare la algoritmii probabilişti (a se vedea şi [2], pp. 141, 147).

Definiţia 7.2. *Un algoritm probabilist este un algoritm care foloseşte numere aleatoare.*

Probabilitatea de succes a unui algoritm probabilist este probabilitatea ca acesta să ofere un răspuns corect.

Fie $0 \leq \epsilon < 1$. Un algoritm tip **Las-Vegas** *este un algoritm probabilist care – pentru orice apariţie a unei probleme – poate oferi un răspuns întotdeauna corect, sau poate eşua şi să nu dea nici un răspuns, cu probabilitatea ϵ.*
Probabilitatea de succes pentru un algoritm Las-Vegas este $1 - \epsilon$.

Observaţia 7.2. *Dacă construim un algoritm Las-Vegas pentru rezolvarea unei probleme, putem să îl apelăm de un număr mediu de $\frac{1}{1-\epsilon}$ ori până se obţine răspuns – care este întotdeauna corect.*

Teorema 7.1. *Fie q un număr prim mare. Următoarele afirmaţii sunt echivalente:*

1. *Există $0 < P_1 \leq 1$ şi un algoritm probabilist polinomial în timp A_1 care – având la intrare un tuplu generator de lungime k şi un element $\alpha \in Z_q^*$ – calculează o reprezentare a lui α cu probabilitate de succes cel puţin P_1.*

2. *Există $0 < P_2 \leq 1$ şi un algoritm probabilist polinomial în timp A_2 care – având la intrare un tuplu generator de lungime k – calculează o reprezentare netrivială a lui 1 cu probabilitate de succes cel puţin P_2.*

3. *Există $0 < P_3 \leq 1$, $\alpha \in Z_q^*$ şi un algoritm probabilist polinomial în timp A_3 care – având la intrare un tuplu generator de lungime k – calculează o reprezentare a lui α cu probabilitate de succes cel puţin P_3.*

4. *Există $0 < P_4 \leq 1$ şi un algoritm probabilist polinomial în timp A_4 care rezolvă problema logaritmilor discreţi (având la intrare $g \in Z_q \setminus \{0, 1\}$ şi $\alpha \in Z_q^*$, calculează $log_g \alpha$) cu probabilitate de succes cel puţin P_4.*

Demonstraţie. Vom demonstra implicaţiile

$$(1) \Longrightarrow (2) \Longrightarrow (4) \Longrightarrow (1) \quad \text{şi} \quad (1) \Longrightarrow (3) \Longrightarrow (4).$$

- $(1) \Longrightarrow (2)$;
  Presupunând existenţa algoritmului A_1 şi a lui P_1, fie $(g_1, \ldots, g_k)$ intrarea în algoritmul A_2. Se introduc în A_1 tuplul generator $(g_1, \ldots, g_k)$ şi $\alpha = 1$. După executarea lui A_1, suntem în posesia unei reprezentări a lui 1 în raport cu $(g_1, \ldots, g_k)$ cu probabilitatea de succes $P_2 = P_1$.

- $(1) \Longrightarrow (3)$;
  Alegem un element arbitrar $\alpha \in Z_q^*$ şi tuplul generator $(g_1, \ldots, g_k)$, intrare pentru A_3. Se execută A_1 care calculează o reprezentare a lui α cu probabilitatea de succes P_1. Atunci există $P_3 = P_1$.

- $(4) \Longrightarrow (1)$;
  Există algoritmul A_4 şi probabilitatea $0 < P_4 \leq 1$. Construim algoritmul A_1 care primeşte la intrare tuplul generator $(g_1, \ldots, g_k)$ şi $\alpha \in Z_q^*$.

1. **for** $i = 1$ **to** $k-1$ **do**
   (a) Se generează aleator $h_i \in Z_q^*$;
   (b) Se introduc g_i, h_i în A_4 care calculează a_i.
   (c) **if** $g_i^{a_i} \neq h_i \ (mod\ q)$ **then** STOP (eşec).
2. Se calculează $a_k = log_{g_k}\left(\alpha \prod_{i=1}^{k-1} h_i^{-1}\right) \ (mod\ q)$;
3. STOP cu succes şi răspuns $(a_1, \ldots, a_k)$.

Corectitudine: Algoritmul astfel construit este de tip Las-Vegas cu probabilitatea de succes $P_1 = \dfrac{P_4}{k-1}$.

În cazul în care A_1 se termină cu succes, la pasul 2 avem $g_i^{a_i} = h_i \ (mod\ q)$, $1 \leq i \leq k-1$ (deci $a_i = log_{g_i} h_i \ (mod\ q)$, $\forall i \ (1 \leq i \leq k-1)$). Urmează

$$a_k = log_{g_k}\left(\alpha \prod_{i=1}^{k-1} h_i^{-1}\right) \ (mod\ q) \iff g_k^{a_k} = \alpha \prod_{i=1}^{k-1} h_i^{-1} \ (mod\ q) \iff$$

$$\iff \prod_{i=1}^{k-1} h_i g_k^{a_k} = \alpha \ (mod\ q) \iff \prod_{i=1}^{k} g_i^{a_i} = \alpha \ (mod\ q) \iff$$

$\iff$ $(a_1, \ldots, a_k)$ este o reprezentare a lui α.

Algoritmul se repetă de un număr mediu de $1/P_4$ ori pentru fiecare $i \in \{1, \ldots, k-1\}$ pentru ca în pasul (b) să obţinem $a_i = log_{g_i} h_i \ (mod\ q)$.
Deci, algoritmul trebuie rulat de un număr mediu de $(k-1)/P_4$ ori, ceea ce înseamnă că probabilitatea de succes a algoritmului A_1 este $P_1 = \dfrac{P_4}{k-1}$.

- $(2) \Longrightarrow (4)$;
  Există algoritmul A_2 şi probabilitatea $0 < P_2 \leq 1$. Construim algoritmul A_4 care primeşte la intrare $g \in Z_q \setminus \{0, 1\}$ şi $\alpha \in Z_q^*$.

1. Se generează aleator un k-tuplu $(u_1, \ldots, u_k) \in Z_q^k$; se calculează
$$g_1 = h^{u_1} \ (mod\ q), \qquad g_i = g^{u_i} \ (mod\ q), \quad (2 \leq i \leq k-1)$$
2. Se generează aleator o permutare $\pi \in S_k$ şi se rulează algoritmul A_2 cu intrarea $(g_{\pi(1)}, \ldots, g_{\pi(k)})$.
3. Fie $(a_1, \ldots, a_k)$ ieşirea lui A_2.
Dacă $\prod_{i=1}^{k} g_{\pi(i)}^{a_i} \neq 1 \ (mod\ q)$ sau $a_{\pi^{-1}(1)} = 0$ atunci STOP (eşec).
4. Se calculează
$$log_g \alpha = -(u_1 a_{\pi^{-1}(1)})^{-1} \sum_{i=2}^{k} u_i a_{\pi^{-1}(i)} \quad (mod\ q)$$
STOP (succes).

Corectitudine: Algoritmul asfel construit este de tip Las Vegas cu probabilitate de succes $P_4 = 2P_2/k$.

Dacă algoritmul se termină cu succes, atunci $(a_1, \ldots, a_k)$ este o reprezentare a lui 1 în raport cu $(g_{\pi(1)}, \ldots, g_{\pi(k)})$, ceea ce este echivalent cu:

$$\prod_{i=1}^{k} g_{\pi(i)}^{a_i} = 1 \ (mod\ q) \iff g_1^{a_{\pi^{-1}(1)}} \prod_{i=2}^{k} g_i^{a_{\pi^{-1}(i)}} = g^0 \ (mod\ q) \iff$$

$$\iff \alpha^{u_1 a_{\pi^{-1}(1)}} \prod_{i=2}^{k} g_{u_i a_{\pi^{-1}(i)}} = g^0 \ (mod\ q)$$

Logaritmând în baza g, relaţia este echivalentă cu

$$u_1 a_{\pi^{-1}(1)} log_g \alpha + \sum_{i=2}^{k} u_i a_{\pi^{-1}(i)} = 0 \quad (mod\ q)$$

de unde se obţine formula din pasul 4 pentru $log_g \alpha$.

De remarcat că la pasul 4, $a_{\pi^{-1}(1)} \neq 0$, ceea ce asigură existenţa inversului $(u_1 a_{\pi - 1(1)})^{-1}$ în Z_q.

Algoritmul se repetă de un număr mediu de $1/P_2$ ori până se obţine la pasul 3 o reprezentare $(a_1, \ldots, a_k)$ a lui 1.

Pentru fiecare reprezentare $(a_1, \ldots, a_k)$ a lui 1, în cazul cel mai defavorabil $k-2$ componente ale k-tuplului $(a_1, \ldots, a_k)$ sunt nule (cum $\prod_{i=1}^{k} g_{\pi(i)}^{a_i} = 1 \ (mod\ q)$, dacă

ar fi nule $k-1$ componente, ar rezulta $a_i = 0\ \forall i = 1, 2, \ldots, k$, deci A_2 ar calcula o reprezentare trivială a lui 1, contradicţie cu ipoteza (2)).
Rezultă $a_{\pi^{-1}(1)} \neq 0$ cu probabilitate cel puţin $2/k$.

Atunci algoritmul se va repeta de cel mult $\frac{1}{P_2} \cdot \frac{k}{2}$ ori, probabilitatea de succes fiind cel puţin $P_4 = 2P_2/k$.

- (3) $\Longrightarrow$ (4);
Există $\alpha' \in Z_q^*$ şi algoritmul A_3 care – având la intrare un tuplu generator de lungime k – determină o reprezentare a lui α' cu probabilitatea de succes P_3.
Construim algoritmul A_4 care primeşte la intrare $g \in Z_q \setminus \{0, 1\}$ şi $\alpha \in Z_q^*$.

1. $i = 1$.
2. Se generează aleator un k-tuplu $(u_1, \ldots, u_k) \in Z_q^k$;
   se calculează (pentru $j = 2, \ldots, k$):
   $$g_{i,1} = \alpha^{u_{i,1}}, \ \ g_{i_j} = g^{u_{i,j}} \quad (mod\ q)$$
3. Se generează aleator o permutare $\pi_i \in S_k$;
   Se rulează algoritmul A_3 cu intrarea $(g_{i,1}, \ldots, g_{i,\pi_i(k)})$.
4. Fie $(a_{i,1}, \ldots, a_{i,k})$ ieşirea lui A_3;
   Dacă $(a_{i,1}, \ldots, a_{i,k})$ nu e o reprezentare a lui α în raport cu $(g_{i,1}, \ldots, g_{i,k})$ atunci STOP (eşec).
5. $i = i + 1$ şi salt la 2.
6. Dacă $u_{2,1}a_{2,\pi_2^{-1}(1)} = u_{1,1}a_{1,\pi_1^{-1}(1)}\ (mod\ q)$ atunci STOP (eşec).
7. Se calculează (modulo q):
   $$log_g\alpha = \left(u_{2,1}a_{2,\pi_2^{-1}(1)} - u_{1,1}a_{1,\pi_1^{-1}(1)}\right)^{-1} \cdot \left(\sum_{j=2}^{k} u_{1,j}a_{1,\pi_1^{-1}(j)} - \sum_{j=2}^{k} u_{2,j}a_{2,\pi_2^{-1}(j)}\right)$$

Corectitudine: Algoritmul asfel construit este de tip Las Vegas cu probabilitatea de succes P_3^2/k.

Dacă algoritmul se termină cu succes, atunci $(a_{1,1}, \ldots, a_{1,k})$ şi $(a_{2,1}, \ldots, a_{2,k})$ sunt reprezentări ale lui α' în raport cu $(g_{1,\pi_1(1)}, \ldots, g_{1,\pi_1(k)})$, respectiv $(g_{2,\pi_1(1)}, \ldots, g_{2,\pi_1(k)})$ ceea ce este echivalent cu

$$\alpha' = \prod_{j=1}^{k} g_{1,\pi_1(j)}^{a_{1,j}} = \prod_{j=1}^{k} g_{2,\pi_2(j)}^{a_{2,j}} \ (mod\ q) \quad \Longleftrightarrow$$

$$\Longleftrightarrow \quad g_{1,1}^{a_{1,\pi_1^{-1}(1)}} \prod_{j=2}^{k} g_{1,j}^{a_{1,\pi_1^{-1}(j)}} = g_{2,1}^{a_{2,\pi_2^{-1}(1)}} \prod_{j=2}^{k} g_{2,j}^{a_{2,\pi_2^{-1}(j)}} \ (mod\ q) \quad \Longleftrightarrow$$

$$\Longleftrightarrow \quad \alpha^{u_{1,1}a_{1,\pi_1^{-1}(1)}} \prod_{j=2}^{k} g^{u_{1,j}a_{1,\pi_1^{-1}(j)}} = \alpha^{u_{2,1}a_{2,\pi_2^{-1}(1)}} \prod_{j=2}^{k} g^{u_{2,j}a_{2,\pi_2^{-1}(j)}} \ (mod\ q)$$

şi logaritmând în baza g, rezultă:

$$u_{1,1}a_{1,\pi_1^{-1}(1)} log_g\alpha + \sum_{j=2}^{k} u_{1,j}a_{1,\pi_1^{-1}(j)} = u_{2,1}a_{2,\pi_2^{-1}(1)} log_g\alpha + \sum_{j=2}^{k} u_{2,j}a_{2,\pi_2^{-1}(j)} \quad (mod\ q)$$

de unde se obţine formula din pasul 7 pentru $log_g\alpha$.

De remarcat că la pasul 7, $u_{1,1}a_{1,\pi_1^{-1}(1)} \neq u_{2,1}a_{1,\pi_2^{-1}(1)}$, ceea ce asigură existenţa inversului $\left(u_{1,1}a_{1,\pi_1^{-1}(1)} - u_{2,1}a_{2,\pi_2^{-1}(1)}\right)^{-1}$ în Z_q.

Pentru $i = 1$ algoritmul se repetă de un număr mediu de $1/P_3$ ori pentru a obţine o reprezentare $(a_{1,1}, \ldots, a_{1,k})$ a lui α'.

Pentru $i = 2$ algoritmul se repetă în cazul cel mai defavorabil de un număr de k/P_3 ori, deoarece trebuie repetat pasul 6 de cel mult k ori.

Astfel, algoritmul se repetă de cel mult k/P_3^2 ori până se obţine răspunsul *"succes"*. Deci probabilitatea de succes este cel puţin P_3^2/k.

Cu această teoremă, echivalenţa celor două probleme din punct de vedere al complexităţii este complet demonstrată. □

Deoarece problema logaritmului discret este $\mathcal{NP}$ - completă în Z_q pentru valori mari ale lui q prim, va rezulta că şi problema reprezentării în Z_q este $\mathcal{NP}$ - completă.

Corolarul 7.1. *Fie q număr prim pentru care problema logaritmului discret este $\mathcal{NP}$ - completă. Atunci nu există un algoritm în timp polinomial care – având la intrare un tuplu generator $(g_1, \ldots, g_k)$ – scoate, cu o probabilitate semnificativă, un număr $\alpha \in Z_q^*$ şi două reprezentări diferite ale lui α în raport cu $(g_1, \ldots, g_k)$.*

Demonstraţie. Prin absurd, presupunem existenţa unui astfel de algoritm A.
Construim următorul algoritm:

1. Fie $(g_1, \ldots, g_k)$ un tuplu generator. Se rulează algoritmul A cu intrarea $(g_1, \ldots, g_k)$.

   Se primesc ca răspuns α şi două reprezentări ale acestuia $(a_1, \ldots, a_k)$ şi $(b_1, \ldots, b_k)$ în raport cu $(g_1, \ldots, g_k)$.

2. Răspuns la ieşire: $(a_1 - b_1\ (mod\ q), \ldots, a_k - b_k\ (mod\ q))$.

Evident: $\prod_{i=1}^{k} g_i^{a_i - b_i} = 1 \ (mod\ q)$, deci

$$(a_1 - b_1 \ (mod\ q), \ldots, a_k - b_k \ (mod\ q))$$

este o reprezentare a lui 1 în raport cu $(g_1, \ldots, g_k)$.

Am construit astfel algoritmul A_2 – polinomial în timp – din Teorema 7.1, ceea ce contrazice presupunerea de $\mathcal{NP}$ - completitudine a problemei logaritmului discret. □

Corolarul 7.2. *Funcţia* $f : Z_q^k \longrightarrow Z_q$ *definită*

$$f(a_1, \ldots, a_k) = \prod_{i=1}^{k} g_i^{a_i}$$

este o funcţie de dispersie criptografică.

Demonstraţie. Din Corolarul 7.1, este calculabil dificil să se obţină un număr $\alpha \in Z_q$ şi două reprezentări diferite ale acestuia $(a_1, \ldots, a_k)$ şi $(b_1, \ldots, b_k)$ în raport cu $(g_1, \ldots, g_k)$. Rezultă că este calculabil dificil să obţinem $x = (a_1, \ldots, a_k) \in Z_q^k$ şi $x_1 = (b_1, \ldots, b_k) \in Z_q^k$ asfel încât $f(x) = f(x_1)$; deci f este o funcţie cu coliziuni tari (criptografică). □

Această funcţie de dispersie nu este suficient de rapidă pentru aplicaţii practice, dar constituie un model teoretic sugestiv.

7.5.3 Demonstrarea cunoaşterii unei reprezentări

Fie p şi q două numere prime mari, cu proprietatea $q|p-1$.

Presupunem că un utilizator *Alice* (*Prover*) cunoaşte o reprezentare $(a_1, \ldots, a_k) \in Z_q^k$ a unui număr $\alpha \in Z_q^*$ în raport cu tuplul generator $(g_1, \ldots, g_k) \in Z_q^k$. *Alice* doreşte să îi demonstreze lui *Bob* (*Verifier*) acest lucru, fără a-i revela reprezentarea sa secretă $(a_1, \ldots, a_k)$.

Construim un protocol de tip **provocare - răspuns** (asemănător cu protocolul Schnorr din secţiunea 8.3) care îi permite lui *Alice* demonstrarea cunoaşterii reprezentării $(a_1, \ldots, a_k)$.

1. *Alice* generează aleator k numere $w_1, \ldots, w_k \in Z_q^*$ şi trimite lui *Bob*

$$z = \prod_{i=1}^{k} g_i^{w_i} \pmod q$$

2. *Bob* generează o provocare $c \in Z_q$ şi o trimite lui *Alice*.

3. *Alice* calculează răspunsul

$$r_i = w_i + ca_i \pmod q, \quad (1 \leq i \leq k)$$

şi trimite $(r_1, \ldots, r_k)$ lui *Bob*.

4. *Bob* acceptă dacă şi numai dacă

$$z\alpha^c = \prod_{i=1}^{k} g_i^{r_i} \pmod q$$

Teorema 7.2.

1. *(***Completitudinea***) Dacă Alice este în posesia reprezentării $(a_1, \ldots, a_k)$ a lui α în raport cu $(g_1, \ldots, g_k)$ şi urmează protocolul pas cu pas, atunci Bob acceptă comunicarea în pasul* 4.

2. *(***Consistenţa***) Dacă Alice nu cunoaşte o reprezentare a lui α în raport cu $(g_1, \ldots, g_k)$, atunci nu există o strategie pentru ea, astfel încât Bob să accepte comunicarea cu o probabilitate semnificativă.*

3. *(***Ascunderea informaţiei***) Chiar dacă Bob ar dispune de o putere de calcul nelimitată, executând protocolul pas cu pas, el nu va reuşi să determine – cu probabilitate semnificativă – reprezentarea cunoscută de Alice.*

Demonstraţie.

1. Presupunând că protocolul este executat corect de ambele părţi, la pasul 4 avem:

$$z\alpha^c = \prod_{i=1}^{k} g_i^{w_i} \alpha^c = \prod_{i=1}^{k} g_i^{w_i} \left(\prod_{i=1}^{k} g_i^{a_i} \right)^c = \prod_{i=1}^{k} g_i^{w_i + a_i c} = \prod_{i=1}^{k} g_i^{r_i} \pmod q$$

deci *Bob* acceptă la pasul 4.

2. *Alice* nu cunoaşte o reprezentare a lui α în raport cu $(g_1, \ldots, g_k)$, deci nu poate să respecte protocolul la pasul 3.
Distingem două situaţii:

   - *Alice* urmează protocolul în pasul 1, deci *Alice* alege aleator $w_1, \ldots, w_k \in Z_q$ şi calculează $z = \prod_{i=1}^{k} g_i^{w_i} \ (mod\ q)$.
   După ce primeşte provocarea c a lui *Bob*, *Alice* trebuie să determine numerele $r_1, \ldots, r_k$ care să verifice relaţia din pasul 4:

$$z\alpha^c = \prod_{i=1}^{k} g_i^{r_i} \ (mod\ p) \quad \Longrightarrow \quad \prod_{i=1}^{k} g_i^{w_i}\alpha^c = \prod_{i=1}^{k} g_i^{r_i} \ (mod\ q) \quad \Longrightarrow$$
$$\Longrightarrow \quad \alpha^c = \prod_{i=1}^{k} g_i^{r_i - w_i} \ (mod\ q) \quad \Longrightarrow \quad \alpha = \prod_{i=1}^{k} g_i^{(r_i - w_i)c^{-1}} \ (mod\ q)$$

   Deci, dacă *Alice* reuşeşte să determine $r_1, \ldots, r_k$ astfel încât *Bob* să accepte comunicarea la pasul 4, el va cunoaşte o reprezentare a lui α în raport cu $(g_1, \ldots, g_k)$ şi anume

$$((r_1 - w_1)c^{-1}, \ldots, (r_k - w_k)c^{-1}),$$

   ceea ce contrazice ipoteza.

   - *Alice* nu urmează protocolul din pasul 1; deci ea va alege un număr z pentru care nu cunoaşte o reprezentare în raport cu $(g_1, \ldots, g_k)$.
   În pasul 3, *Alice* trebuie să determine numerele $r_1, \ldots, r_k$ astfel încât

$$z\alpha^c = \prod_{i=1}^{k} g_i^{r_i} \quad (mod\ q),$$

   problemă echivalentă cu determinarea unei reprezentări a lui $z\alpha^c$ în raport cu $(g_1, \ldots, g_k)$.
   Cum problema reprezentării este $\mathcal{NP}$ - completă pentru p, q convenabil alese şi protocolul se desfăşoară în timp real, nici în acest caz *Alice* nu are o strategie – cu o probabilitate semnificativă de succes – pentru a-l convinge pe *Bob* să accepte comunicarea.

3. La sfârşitul protocolului *Bob* este în posesia următoarelor informaţii: $z, c, r_1, \ldots, r_k$. Avem

$$r_i = w_i + ca_i \quad (mod\ q) \quad \Longleftrightarrow \quad a_i = c^{-1}(r_i - w_i) \quad (mod\ q),$$

   deci problema determinării lui $(a_1, \ldots, a_k)$ este echivalentă cu problema determinării lui $(w_1, \ldots, w_k)$.
   În plus $a_1, \ldots, a_k$ sunt unic determinate de $w_1, \ldots, w_k$.

Singura informaţie pe care *Bob* o are despre $w_1, \ldots, w_k$ este $z = \prod_{i=1}^{k} g_i^{w_i} \ (mod\ q)$.

Dispunând de o putere de calcul nelimitată, *Bob* poate să determine o reprezentare a lui z în raport cu $(g_1, \ldots, g_k)$.

Conform Propozitiei 7.2, există exact q^{k-1} reprezentări ale lui z în raport cu $(g_1, \ldots, g_k)$, deci probabilitatea ca *Bob* să calculeze exact aceeaşi reprezentare $(w_1, \ldots, w_k)$ este $\frac{1}{q^{k-1}}$, adică neglijabilă.

□

7.6 Sistemul electronic de plată Brands

Cele mai cunoscute şi utilizate protocoale de plată "off-line" cu monede electronice aparţin lui Ştefan Brands ([11]). Primul sistem real de plată "off-line", numit "*DigiCash*", se bazează în totalitate pe aceste protocoale.

Schema Brands este considerată cea mai bună construcţie funcţională, din cel puţin două motive:

- Nu utilizează tehnica Cut - and - Choose (protocol destul de greoi din punct de vedere al dimensiunilor).
- Se bazează în totalitate pe protocolul Schnorr, ceea ce permite diverse medii de implementare, în particular pe curbe eliptice.

7.6.1 Iniţializarea sistemului Brands

Fie p, q numere prime mari cu proprietatea $q|p-1$ (similar schemelor de autentificare din secţiunea 7.2.) şi H o funcţie de dispersie criptografică.
Fie G_q grupul multiplicativ generat de q.

Iniţializarea sistemului constă în generarea de către bancă a următoarelor cinci numere aleatoare distincte:

1. $g \in G_q,\ x \in Z_q^*$; se calculează $h = g^x \ (mod\ q)$.
   Perechea $(g, h) \in G_q \times G_q$ constituie cheia publică a băncii, iar $x \in Z_q^*$ cheia secretă.
2. O pereche (g_1, g_2) cu $g_1, g_2 \in G_q$.
3. Un generator $d \in G_q$.

În prima fază vom presupune că există doar monede de o singură valoare; vom arăta ulterior cum se pot introduce monede de valori diferite.

Securitatea acestei scheme constă în alegerea numerelor g, h, g_1, g_2, d astfel încât nici un utilizator să nu poată exprima oricare din aceste numere ca o combinaţie de puteri ale celorlalte. Acest lucru este din punct de vedere al complexităţii la fel de dificil ca şi găsirea unei reprezentări netriviale a lui 1 în raport cu (g, h, g_1, g_2, d) – o problemă $\mathcal{NP}$ - completă pentru p şi q convenabil alese (secţiunea 7.5.).

Când un utilizator $\mathcal{U}$ îşi deschide un cont în banca $\mathcal{B}$, el generează aleator numerele $u_1, u_2 \in Z_q^*$ şi calculează

$$I = g_1^{u_1} g_2^{u_2} \pmod p$$

(u_1, u_2) va constitui cheia secretă a lui $\mathcal{U}$, iar $I \in G_q$ va fi pseudonimul folosit de $\mathcal{B}$ pentru identitatea reală a lui $\mathcal{U}$.

$\mathcal{B}$ stochează într-o bază de date identitatea reală a lui $\mathcal{U}$, numărul contului său şi I.

Este important ca $\mathcal{U}$ să cunoască o singură reprezentare a lui I în raport cu (g_1, g_2). Dacă $\mathcal{U}$ utilizează o monedă de două ori, $\mathcal{B}$ poate să determine (u_1, u_2).

O monedă electronică este reprezentată sub forma unui mesaj semnat – $(m, sig_K(m))$ cu anumite restricţii asupra lui m.

7.6.2 Tehnici pentru crearea sistemului Brands

Protocolul de semnătură Chaum - Pedersen

Protocolul de semnătură Chaum - Pedersen este un protocol de tip Schnorr.
Fie $m \in G_q$ mesajul care trebuie semnat. Presupunem că $\mathcal{U}$ doreşte o semnătură a lui $\mathcal{B}$ pentru mesajul m. Paşii protocolului sunt următorii:

1. $\mathcal{B}$ generează aleator $w \in Z_q$ şi trimite lui $\mathcal{U}$ numerele

$$z = m^x \pmod p, \quad a = g^w \pmod p, \quad b = m^w \pmod p$$

2. $\mathcal{U}$ generează aleator provocarea $c = H(m\|z\|a\|b)$ şi o trimite lui $\mathcal{B}$.

3. $\mathcal{B}$ răspunde cu $r = w + cx \pmod q$.

4. $\mathcal{U}$ acceptă dacă şi numai dacă

$$h^c a = g^r \pmod p, \quad z^c b = m^r \pmod p$$

Se observă că singura diferenţă faţă de protocolul de semnătură Schnorr este trimiterea la pasul 1 a încă două numere: z şi a.

$sig_K(m) = (z, a, b, r)$ va constitui semnătura mesajului m.

O reprezentare formală a protocolului de semnatură Chaum - Pedersen (conformă cu notaţiile din Capitolul 3) este:

$\mathcal{P} = G_q, \quad \mathcal{A} = G_p^3 \times G_q, \quad \mathcal{K} = \{(p,q,g,h,x) |\ p,q \text{ prime}, q|p-1, h = g^x \ (mod\ p)\}.$
Cheia (p,q,g,h) este publică, iar x este secretă.
Pentru o cheie $K \in \mathcal{K}$, un mesaj $m \in G_q$ şi un număr aleator $w \in Z_q$, definim

$$sig_K(m,w) = (z,a,b,r),$$

unde
$z = m^x \ (mod\ p), \quad a = g^w \ (mod\ p), \quad b = m^w \ (mod\ p), \quad c = H(m\|z\|a\|b),$
$r = w + cx \ (mod\ q)$
Procedura de verificare este:

$$ver_K(m,(z,a,b,r)) = T \quad \Longleftrightarrow \quad h^c a = g^r \ (mod\ p), \quad z^c b = m^r \ (mod\ p)$$

Dacă $\mathcal{U}$ şi $\mathcal{B}$ urmează protocolul, după primii 3 paşi avem:

$$g^r = g^{w+cx} = (g^x)^c g^w = h^c a \ (mod\ p),$$
$$m^r = m^{w+cx} = (m^x)^c m^w = z^c b \ (mod\ p)$$

deci la pasul 4, $\mathcal{U}$ acceptă comunicarea.

Protocolul restrictiv de semnătură blind Chaum - Pedersen

Vom descrie o variantă simplificată a schemei de semnătură blind folosită în protocolul de extragere a monedelor din bancă.
Pentru a transforma protocolul de semnătură Chaum - Pedersen într-un protocol de semnătură blind, $\mathcal{U}$ trebuie să obţină un nou mesaj m' şi noile numere $a',\ b',\ z',\ c' = H(m'\|z'\|a'\|b')$.
$\mathcal{B}$ răspunde la provocare cu valoarea r, pe care $\mathcal{U}$ o transformă în r' astfel încât (z',a',b',r') să fie o semnătură validă a lui m'.

1. $\mathcal{B}$ generează aleator $w \in Z_q$ şi trimite lui $\mathcal{U}$ valorile

$$z = m^x \ (mod\ p), \quad a = g^w \ (mod\ p), \quad b = m^w \ (mod\ p)$$

2. $\mathcal{U}$ generează aleator trei numere $s \in Z_q^*,\ u,v \in Z_q$, apoi calculează:

$m' = m^s \ (mod\ p)$, $\quad a' = a^u g^v = g^{w'} \ (mod\ p)$,
$b' = a^{us}(m')^v = (m')^{w'} \ (mod\ p)$, $\quad z' = z^s = (m')^x \ (mod\ p)$,
$c' = H(m'||z'||a'||b')$ $\quad$ (s-a notat $w' = uw + v \ (mod\ q)$)

şi trimite lui $\mathcal{B}$ numărul $c = c'u^{-1} \ (mod\ q)$.

3. $\mathcal{B}$ răspunde cu $r = w + cx \ (mod\ q)$.

4. $\mathcal{U}$ acceptă dacă şi numai dacă

$$h^c a = g^r \ (mod\ p), \qquad z^c b = m^r \ (mod\ p)$$

Observaţia 7.3. *Se observă că (z, a, b, r) nu este o semnătură validă asupra lui m deoarece r este un răspuns la o provocare diferită de $H(m\|z\|a\|b)$.*
Dar $\mathcal{U}$ poate calcula $r' = ur + v \ (mod\ q)$ şi obţine $sig_K(m') = (z', a', b', r')$, o semnătură validă asupra lui m'.
Semnătura poate fi verificată de oricine.

$\mathcal{U}$ acceptă la pasul 4 deoarece relaţiile sunt verificate la fel ca în protocolul anterior.

Lema 7.3. *Dacă $r' = ur + v \ (mod\ q)$, atunci (z', a', b', r') constituie o semnătură validă asupra lui m'.*

Demonstraţie. Pentru o cheie $K = (p, q, g, h, x, w)$, procedura de verificare este

$$ver_K(m', (z', a', b', r')) = T \iff \begin{cases} h^{c'} a' = g^{r'} \ (mod\ p) \\ (z')^{c'} b' = (m')^{r'} \ (mod\ p) \end{cases}$$

Dar $r' = ur + v = u(w + cx) + v \ (mod\ q) \implies$
$\implies g^{r'} = g^{u(w+cx)+v} = g^{ucx} g^{v+uw} = (g^x)^{uc} g^{w'} = h^{c'} a' (mod\ p)$
şi $(m')^{r'} = (m')^{ucx} (m')^{uw+v} = (m')^{xc'} (m')^{w'} = (z')^{c'} b' (mod\ p)$,
deci $sig_K(m') = (z', a', b', r')$. □

Protocolul este de tip "blind" deoarece $\mathcal{B}$ nu cunoaşte m', mesajul pentru care $\mathcal{U}$ a obţinut o semnătură.

Lema 7.4.

1. *Dacă $\mathcal{U}$ urmează protocolul pas cu pas, atunci perechea $(m', sig_K(m'))$ nu poate fi legată de nici o execuţie specifică a protocolului.*

2. *Chiar dacă $\mathcal{B}$ află m', reprezentarea (s, t) a lui m' în raport cu (m, g) cunoscută de $\mathcal{U}$ îi este ascunsă lui $\mathcal{B}$ în setul tuturor reprezentărilor lui m'.*

Demonstraţie. La finalul protocolului, $\mathcal{B}$ deţine următoarele informaţii: w, z, a, b, c.
Arătăm că există q alegeri posibile pentru (s, u, v) care produc în final aceeaşi pereche $(m', sig_K(m'))$.

Fie $s \in Z_q^*$ fixat (sunt posibile $q - 1$ alegeri pentru s) şi $m', sig_K(m') = (z', a', b', r')$ fixate.
Să analizăm în câte moduri putem alege valorile u, v.

Se calculează (în mod unic) amprenta $c' = H(m'\|z'\|a'\|b') \ (mod\ q)$. Deci
$uc = c'(mod\ q) \quad \Longrightarrow \quad u = c'c^{-1} \ (mod\ q)$, de unde rezultă că şi u este unic determinat.

$r' = ur + v = u(w + cx) + v = uw + c'x + v \ (mod\ q) \quad \Longrightarrow \quad v = r' - uw - c'x \ (mod\ q)$.
Să arătăm că valorile $u,\ v$ astfel determinate verifică relaţiile pentru a' şi b'.

$a^u g^v = a^u g^{r'-uw-c'x} = a^u g^{r'}(g^w)^{-u}(g^x)^{-c'} = a^u h^{c'} a' a^{-u} h^{-c'} = a'(mod\ p)$

$(m')^{w'} = (m')^{uv+w} = b^{us}(m')^v = b^{us}(m')^{r'-uw-c'x} = (m^w)^{us}(m')^{r'}(m')^{-uw}((m')^x)^{-c'} = m^{wus}(z')^{c'} b'(m^s)^{-uw}(z')^{-c'} = b'(mod\ p)$

Am demonstrat astfel că există exact $q-1$ alegeri pentru (s, u, v) care produc aceeaşi pereche $(m', sig_K(m'))$, deci $\mathcal{B}$ nu realizează în care execuţie specifică a protocolului s-a semnat mesajul m'. □

Generarea aleatoare a unei reprezentari

În faza de iniţializare a sistemului de plată Brands, $\mathcal{U}$ trebuie să genereze un tuplu (u_1, u_2) pentru a-şi deschide un cont în banca $\mathcal{B}$ şi a calcula pseudonimul său $I = g_1^{u_1} g_2^{u_2} \ (mod\ p)$. Banca trebuie să se asigure că $\mathcal{U}$ nu cunoaşte două reprezentări ale lui I în raport cu (g_1, g_2).

Prin următorul protocol, $\mathcal{U}$ şi $\mathcal{B}$ generează împreună un număr aleator $h \in Z_q^*$ şi o reprezentare a acestuia în raport cu tuplul generator $(g_1, \ldots, g_k)$.

1. $\mathcal{U}$ generează aleator tuplul $(x_1, \ldots, x_{k+1})$ şi trimite lui $\mathcal{B}$
$$h' = \prod_{i=1}^{k+1} g_i^{x_i} \quad (mod\ p)$$
2. $\mathcal{B}$ trimite lui $\mathcal{U}$ un tuplu generat aleator $(y_1, \ldots, y_k)$.
3. $\mathcal{U}$ trimite x_{k+1} lui $\mathcal{B}$.

Dacă $\mathcal{U}$ a fost corect la pasul 3, $(x_1 + y_1, \ldots, x_k + y_k)$ este o reprezentare a lui
$$h = h' g_{k+1}^{x_{k+1}} \left(\prod_{i=1}^{k} g_i^{y_i} \right) \quad (mod\ p)$$
în raport cu $(g_1, \ldots, g_k)$. Într-adevăr
$$g_1^{x_1+y_1} \ldots g_k^{x_k+y_k} = \prod_{i=1}^{k} g_i^{x_i} \prod_{i=1}^{k} g_i^{y_i} = h' g_{k+1}^{-x_{k+1}} \prod_{i=1}^{k} g_i^{y_i} = h \quad (mod\ p)$$

7.6.3 Sistemul de bază Brands

În această secţiune sunt descrise cele trei protocoale care formează sistemul de bază Brands.
Se presupune – pentru început – că în sistem există doar monede de o valoare unică.

Protocolul de extragere a monedelor

Dacă utilizatorul $\mathcal{U}$ – având identitatea $I = g_1^{u_1} g_2^{u_2} \ (mod\ p)$ – doreşte să extragă o monedă din contul său, mai întâi trebuie să convingă banca că într-adevăr el este posesorul contului de unde se face extragerea. Pentru aceasta $\mathcal{U}$ şi $\mathcal{B}$ vor executa protocolul provocare / răspuns Schnorr, prin care $\mathcal{U}$ îi demonstrează lui $\mathcal{B}$ cunoaşterea unei reprezentări a lui I în raport cu (g_1, g_2), fără a o releva.

Pentru fiecare extragere a unei monede, este executat următorul protocol:

1. $\mathcal{U}$ îi demonstrează lui $\mathcal{B}$ cunoaşterea reprezentării (u_1, u_2) a lui I în raport cu (g_1, g_2) prin protocolul Schnorr.
2. $\mathcal{B}$ realizează următoarele operaţii:
   (a) extrage din contul lui $\mathcal{U}$ valoarea fixată a monedei;
   (b) calculează $m = I \cdot d \ (mod\ p)$;
   (c) generează aleator $w \in Z_q$;
   (d) trimite lui $\mathcal{U}$: $z = m^x \ (mod\ p)$, $a = g^w \ (mod\ p)$ şi $b = m^w \ (mod\ p)$.
3. $\mathcal{U}$ realizează următoarele operaţii:
   (a) calculează $m = Id \ (mod\ p)$;
   (b) generează aleator $s \in Z_q^*$, $u, v \in Z_q$;
   (c) calculează:
   $m' = m^s = I^s d^s = g_1^{u_1 s} g_2^{u_2 s} d^s \ (mod\ p)$, $z' = z^s \quad (mod\ p)$,
   $w' = uw + v \quad (mod\ q)$, $a' = a^u g^v = g^{w'} \quad (mod\ p)$,
   $b' = b^{us}(m')^v = (m')^{w'} \quad (mod\ p)$

(d) $\mathcal{U}$ determină o descompunere aleatoare a lui m' în A şi B ($m' = A \cdot B \ (mod\ p)$) astfel: determină aleator fiecare din numerele $u_1 s,\ u_2 s$ ca sumă de două numere:

$$u_1 s = x_1 + x_2 \ (mod\ q), \quad u_2 s = y_1 + y_2 \ (mod\ q), \quad s = z_1 + z_2 \ (mod\ q)$$

apoi calculează

$$A = g_1^{x_1} g_2^{y_1} d^{z_1} \ (mod\ p), \qquad B = g_1^{x_2} g_2^{y_2} d^{z_2} \ (mod\ p)$$

(e) calculează $c' = H(m' \| z' \| a' \| b' \| A)$;

(f) trimite lui $\mathcal{B}$ mesajul $c = c' u^{-1} \ (mod\ q)$.

4. $\mathcal{B}$ răspunde cu $r = cx + w \ (mod\ q)$.

5. $\mathcal{U}$ acceptă dacă şi numai dacă

$$g^r = h^c a \ (mod\ p), \qquad m^r = z^c b \ (mod\ p)$$

În final, $\mathcal{U}$ calculează $r' = ru + v \ (mod\ q)$.

Conform Lemelor 7.3 şi 7.4, în final $\mathcal{U}$ este în posesia unui mesaj semnat

$$(A, B, sig_K(A, B)) = (z', a', b', r')$$

care nu poate fi legat de nici o execuţie specifică a protocolului. În plus,

$$m' = A \cdot B \quad (mod\ p)$$

Descompunerea lui m' în A şi B a fost făcută pentru ca $\mathcal{U}$ să poată demonstra – în protocolul de plată – identitatea sa, fără a o dezvălui.

Dacă $\mathcal{U}$ urmează protocolul, atunci se va afla în final în posesia reprezentărilor lui m', A, B în raport cu (g_1, g_2, d) care sunt $(u_1 s, u_2 s, s)$, (x_1, y_1, z_1) şi respectiv (x_2, y_2, z_2). Din Corolarul 7.1, $\mathcal{U}$ nu poate afla două reprezentări ale lui m' în timp polinomial.

Cum $(x_1 + x_2, y_1 + y_2, z_1 + z_2)$ este de asemenea o reprezentare a lui m' rezultă că trebuie îndeplinite simultan condiţiile (toate modulo q):

$$u_1 s = x_1 + x_2, \quad u_2 s = y_1 + y_2, \quad s = z_1 + z_2$$

Faptul că $\mathcal{U}$ acceptă în pasul 5 se deduce similar protocolului de semnătură blind Chaum - Pedersen.

Protocolul de plată

Când $\mathcal{U}$ doreşte să folosească o monedă $(A, B, sig_K(A, B))$ la magazinul $\mathcal{S}$, este executat următorul protocol:

1. $\mathcal{U}$ îi trimite lui $\mathcal{S}$:

$$A = g_1^{x_1} g_2^{y_1} d^{z_1} \ (mod\ p), \quad B = g_1^{x_2} g_2^{y_2} d^{z_2} \ (mod\ p), \quad sig_K(A, B) = (z', a', b', r')$$

2. $\mathcal{S}$ se asigură că $A \cdot B \neq 1$.
   Apoi el verifică semnătura băncii:

$$ver_K(m', sig_K(m')) = T \iff \begin{cases} g^{r'} = h^{c'} a' & (mod\ p) \\ (m')^{r'} = (z')^{c'} b' & (mod\ p) \end{cases} \text{ cu}$$

   $c' = H(A \cdot B \| z' \| a' \| b' \| A)$.

   Dacă relaţiile sunt îndeplinite, $\mathcal{S}$ este convins că moneda a fost emisă de bancă şi trimite lui $\mathcal{U}$ o provocare $c \in Z_q \setminus \{0, 1\}$.

3. $\mathcal{U}$ răspunde cu
   $r_1 = x_1 + cx_2 \ (mod\ q), \quad r_2 = y_1 + cy_2 \ (mod\ q), \quad r_3 = z_1 + cz_2 \ (mod\ q)$

4. $\mathcal{S}$ acceptă dacă şi numai dacă

$$g_1^{r_1} g_2^{r_2} d^{r_3} \equiv A \cdot B^c \quad (mod\ p)$$

Teorema 7.3.

1. *(***Completitudinea***) Dacă $\mathcal{U}$ urmează protocolul pas cu pas şi protocolul de extragere a monedei a fost executat corect, atunci $\mathcal{S}$ acceptă la pasul* 4.
2. *(***Consistenţa***) Dacă $\mathcal{U}$ nu cunoaşte o reprezentare a lui A, respectiv B în raport cu (g_1, g_2, d), atunci nu există strategie pentru el astfel încât $\mathcal{S}$ să accepte cu o probabilitate semnificativă de succes.*
3. *(***Ascunderea informaţiei***) Chiar dacă $\mathcal{S}$ dispune de o putere de calcul nelimitată, iar $\mathcal{U}$ şi $\mathcal{S}$ execută protocolul pas cu pas, probabilitatea ca $\mathcal{S}$ să afle în final identitatea lui $\mathcal{U}$ este neglijabilă.*

Demonstraţie.

1. Dacă protocolul este executat pas cu pas, la pasul 4 avem:

$$g_1^{r_1} g_2^{r_2} d^{r_3} = g_1^{x_1 + cx_2} g_2^{y_1 + cy_2} d^{z_1 + cz_2} = g_1^{x_1} g_2^{y_1} d^{z_1} (g_1^{x_2} g_2^{y_2} d^{z_2})^c = A \cdot B^c \ (mod\ p)$$

   deci $\mathcal{S}$ acceptă.

2. $\mathcal{U}$ ar trebui să determine numerele $r_1, r_2, r_3 \in Z_q$ astfel încât

$$g_1^{r_1} g_2^{r_2} d^{r_3} \equiv A \cdot B^c \pmod{p},$$

ceea ce este echivalent cu determinarea unei reprezentări a lui AB^c în raport cu (g_1, g_2, d), care este o problemă pe care $\mathcal{U}$ nu o poate rezolva în timp polinomial.

Astfel, se deduce un lucru foarte important pentru securitatea sistemului:

$\mathcal{S}$ acceptă $\iff$ $\mathcal{U}$ cunoaşte o reprezentare a lui A şi B în raport cu (g_1, g_2, d) şi $\mathcal{S}$ este cinstit $\implies$ $\mathcal{U}$ cunoaşte o reprezentare a lui m' în raport cu (g_1, g_2, d).

Deci, dacă $\mathcal{U}$ nu cunoaşte o reprezentare a lui m' în raport cu (g_1, g_2, d), nu are nici o şansă să-l facă pe $\mathcal{S}$ să accepte la pasul 4.

3. La sfârşitul protocolului, $\mathcal{S}$ se află în posesia următoarelor informaţii:
$$A, B, z', a', b', r', c, r_1, r_2, r_3.$$

Considerăm sistemul (întâi cu necunoscutele $u_1, u_2, s, r_1, r_2, r_3$, iar a doua oară cu necunoscutele $x_1, x_2, y_1, y_2, z_1, z_2$):

$$\begin{cases} x_1 + x_2 = u_1 s & y_1 + y_2 = u_2 s & z_1 + z_2 = s \\ x_1 + cx_2 = r_1 & y_1 + cy_2 = r_2 & z_1 + cz_2 = r_3 \end{cases}$$

(toate ecuaţiile sunt scrise modulo q).

Din acest sistem se deduce că problema determinării lui (u_1, u_2) este echivalentă cu problema determinării numerelor: $x_1, x_2, y_1, y_2, z_1, z_2$.

Informaţiile pe care $\mathcal{S}$ le are despre $x_1, x_2, y_1, y_2, z_1, z_2$ sunt:

$$\begin{cases} x_1 + cx_2 \equiv r_1 \pmod{q} & (1) \\ y_1 + cy_2 \equiv r_2 \pmod{q} & (2) \\ z_1 + cz_2 \equiv r_3 \pmod{q} & (3) \end{cases} \qquad \begin{cases} A \equiv g_1^{x_1} g_2^{y_1} d^{z_1} \pmod{p} & (4) \\ B \equiv g_1^{x_2} g_2^{y_2} d^{z_2} \pmod{p} & (5) \\ g_1^{r_1} g_2^{r_2} d^{r_3} \equiv AB^c \pmod{p} & (6) \end{cases}$$

$(1) \iff x_1 \equiv r_1 - cx_2 \pmod{q}$,
$(2) \iff y_1 \equiv r_2 - cy_2 \pmod{q}$,
$(3) \iff z_1 \equiv r_3 - cz_2 \pmod{q}$

Înlocuind x_1, y_1, z_1 în (4) şi folosind (6) rezultă:

$A \equiv g_1^{r_1 - cx_2} g_2^{r_2 - cy_2} d^{r_3 - cz_2} \pmod{p} \iff g_1^{r_1} g_2^{r_2} d^{r_3} \equiv A(g_1^{x_2} g_2^{y_2} d^{z_2})^c \pmod{p} \iff$
$g_1^{x_2} g_2^{y_2} d^{z_2} = (A^{-1} g_1^{r_1} g_2^{r_2} d^{r_3})^{c^{-1}} \equiv B \pmod{p}$. (7)

Relaţia (7) este echivalentă cu (5), ceea ce demonstrează că din relaţiile (1),(2) şi (3) nu se obţine nici o informaţie în plus, deci (4) şi (5) sunt de fapt singurele informaţii esenţiale pe care $\mathcal{S}$ le are despre $x_1, x_2, y_1, y_2, z_1, z_2$.

De aici rezultă că determinarea lui (x_1, y_1, z_1) este echivalentă cu determinarea unei reprezentări a lui A în raport cu (g_1, g_2, d), iar determinarea lui (x_2, y_2, z_2) este echivalentă cu determinarea unei reprezentări a lui B.

Dacă dispune de o putere de calcul nelimitată, $\mathcal{S}$ poate determina o reprezentare a lui A şi o reprezentare a lui B în raport cu (g_1, g_2, d). Probabilitatea ca aceste reprezentări să fie egale cu cele cunoscute de $\mathcal{U}$ este $\frac{1}{q^2} \cdot \frac{1}{q^2} = \frac{1}{q^4}$ (conform Lemei 7.2).
Deci probabilitatea ca $\mathcal{S}$ să afle identitatea lui $\mathcal{U}$ după efectuarea protocolului este q^{-4}.

□

Protocolul de depozit

După un anumit interval de timp, toate magazinele depozitează la bancă monedele electronice primite de la clienţi. Protocolul executat de un magazin $\mathcal{S}$ cu banca pentru trimiterea fiecărei monede este următorul:

1. $\mathcal{S}$ şi $\mathcal{B}$ urmează protocolul provocare/răspuns Schnorr prin care $\mathcal{S}$ demonstrează cunoaşterea reprezentării identităţii sale în raport cu (g_1, g_2).
2. $\mathcal{S}$ trimite lui $\mathcal{B}$ mesajele $A, B, sig_K(A, B), c, r_1, r_2, r_3$.
3. $\mathcal{B}$ verifică validitatea semnăturii $sig_K(A, B)$, stochează $A, B, sig_K(A, B), c$, r_1, r_2, r_3 într-o bază de date şi creditează contul lui $\mathcal{S}$ cu valoarea monedei.

Teorema 7.4. *Dacă moneda $(A, B, sig_K(A, B))$ apare de două ori în baza de date a băncii, atunci banca poate determina identitatea utilizatorului care a utilizat moneda de două ori.*

Demonstraţie. Dacă moneda $(A, B, sig_K(A, B))$ apare de două ori în baza de date, rezultă că banca dispune de două seturi de răspunsuri ale lui $\mathcal{U}$: (r_1, r_2, r_3) şi (r_1', r_2', r_3') corespunzătoare celor două provocări c şi c' ale lui $\mathcal{S}$.

Din relaţiile de verificare ale protocolului de plată rezultă:

$$\begin{cases} g_1^{r_1} g_2^{r_2} d^{r_3} \equiv AB^c \ (mod\ p) & (8) \\ g_1^{r_1'} g_2^{r_2'} d^{r_3'} \equiv AB^{c'} \ (mod\ p) & (9) \end{cases}$$

Ridicând (1) la puterea c' şi (2) la puterea c, rezultă:

$$\begin{cases} A^{c'} B^{cc'} \equiv g_1^{r_1 c'} g_2^{r_2 c'} d^{r_3 c'} & (mod\ p) \\ A^{c} B^{cc'} \equiv g_1^{r_1' c} g_2^{r_2' c} d^{r_3' c} & (mod\ p) \end{cases}$$

Împărţind cele două relaţii rezultă:

$A^{c'-c} \equiv g_1^{r_1c'-r_1'c} g_2^{r_2c'-r_2'c} d^{r_3c'-r_3'c} \pmod p \implies$

$A \equiv g_1^{(r_1c'-r_1'c)(c'-c)^{-1}} g_2^{(r_2c'-r_2'c)(c'-c)^{-1}} d^{(r_3c'-r_3'c)(c'-c)^{-1}} \pmod p \quad (10)$

Analog se obţine:

$$B \equiv g_1^{(r_1-r_1')(c-c')^{-1}} g_2^{(r_2-r_2')(c-c')^{-1}} d^{(r_3-r_3')(c-c')^{-1}} \pmod p \quad (11)$$

Dar

$$A \equiv g_1^{x_1} g_2^{y_1} d^{z_1} \pmod p \quad (12)$$

şi

$$B \equiv g_1^{x_2} g_2^{y_2} d^{z_2} \pmod p \quad (13)$$

Cum nu se cunoaşte o reprezentare netrivială a lui 1 în raport cu (g_1, g_2, d), din (10),(11),(12) şi (13) rezultă (toate relaţiile sunt calculate modulo q):

$$\begin{cases} (r_1c' - r_1'c)(c'-c)^{-1} = x_1 & (r_2c' - r_2'c)(c'-c)^{-1} = y_1 & (r_3c' - r_3'c)(c'-c)^{-1} = z_1 \\ (r_1 - r_1')(c-c')^{-1} = x_2 & (r_2 - r_2')(c-c')^{-1} = y_2 & (r_3 - r_3')(c-c')^{-1} = z_2 \end{cases}$$

Ceea ce înseamnă că banca poate să calculeze:

$$u_1 s = x_1 + x_2 = (r_1c' - r_1'c)(c'-c)^{-1} + (r_1 - r_1')(c-c')^{-1} \pmod q \quad (14)$$

$$u_2 s = y_1 + y_2 = (r_2c' - r_2'c)(c'-c)^{-1} + (r_2 - r_2')(c-c')^{-1} = y_2 \pmod q \quad (15)$$

$$s = z_1 + z_2 = (r_3c' - r_3'c)(c'-c)^{-1} + (r_3 - r_3')(c-c')^{-1} = z_2 \pmod q \quad (16)$$

Din (16) rezultă $s = (c'-c)^{-1}(r_3(c'-1) - r_3'(c-1)) \pmod q \implies$

$$s^{-1} = (r_3(c'-1) - r_3'(c-1))^{-1}(c'-c) \pmod q \quad (17).$$

Din (14) şi (17) se obţine

$u_1 = u_1 s s^{-1} = (c'-c)^{-1}(r_1(c'-1) - r_1'(c-1))(r_3(c'-1) - r_3'(c-1))^{-1}(c'-c) = (r_1(c'-1) - r_1'(c-1))(r_3(c'-1) - r_3'(c-1))^{-1} \pmod q$

şi analog

$u_2 = (r_2(c'-1) - r_2'(c-1))(r_3(c'-1) - r_3'(c-1))^{-1} \pmod q$,

deci banca poate să calculeze identitatea utilizatorului $\mathcal{U}$. □

7.6.4 Corectitudinea sistemului de bază

Următoarea teoremă este fundamentală pentru demonstrarea securităţii sistemului:

Teorema 7.5. *În protocolul de extragere a monedelor, $\mathcal{U}$ nu poate să obţină o semnătură asupra unui mesaj m' pentru care cunoaşte o reprezentare în raport cu (g_1, g_2, d), dar care nu este o putere a lui m.*

Demonstraţie. Presupunem – prin reducere la absurd – că $\mathcal{U}$ poate să obţină o semnătură asupra lui m', cu m' de forma

$$m' = m^s g_1^r = g_1^{u_1 s + r} g_2^{u_2 s} d^s \pmod p,$$

cu $s, r \in Z_q$ arbitrare. Celelalte cazuri pentru forma lui m' se vor rezolva analog.

Dacă $\mathcal{U}$ poate obţine o semnătură validă asupra lui m', rezultă că $\mathcal{U}$ poate să calculeze $sig_K(m') = (z', a', b', r')$ în timp polinomial.

Să vedem cât de dificilă este pentru $\mathcal{U}$ calcularea lui b'.

$$b' = (m')^{w'} = (m')^{uw+v} = (m^s g_1^r)^{uw+v} = m^{suw} m^{sv} (g_1^{ru})^w g_1^{rv} = b^{su} m^{sv} g_1^{rv} (g_1^{ru})^w \pmod p$$

Notăm $g_3 = g_1^{ru}$. Deci

$$g_3^w \equiv g_1^{-rv} m^{-sv} b^{-su} b' \pmod q,$$

cu $g_1, m, b, s, u, v, r, g_3$ cunoscute, iar w necunoscut lui $\mathcal{U}$ (w este generat aleator de $\mathcal{B}$ la fiecare execuţie a protocolului).

Deci calcularea lui b' este la fel de dificilă ca şi calcularea lui g_3^w.

Fie $s \in Z_p$ astfel încât $g_3 = g^s \ (mod\ p)$.

Presupunând că b' se poate calcula în timp polinomial, rezultă că şi g_3^w se poate calcula în timp polinomial.

Dar $\mathcal{U}$ cunoaşte $g_3 = g^s, g^w, g, m, m^w$ şi reuşeşte să determine $g^{ws} = g_3^w$, adică o cheie Diffie - Hellman, unică pentru g^w, g^s.

Cum u şi w sunt alese aleator, avem că şi $g^w, g^s = g_3 = g_1^{ru}$ sunt aleatoare în Z_q.

Am construit asfel un algoritm polinomial care are la intrare numerele aleatoare g^s, g^w şi calculează o cheie Diffie - Hellman unică, ceea ce reprezintă o contradicţie cu $\mathcal{NP}$ - completitudinea acestei probleme.

De aici rezultă că b' nu se poate calcula polinomial, deci $\mathcal{U}$ nu poate obţine o semnătură asupra lui m' care nu e multiplu de m. □

Consecinţă: Din Teoremele 7.3 şi 7.5 deducem că $\mathcal{U}$ poate să obţină o semnătură asupra lui m' pe care o poate folosi într-un protocol de plată dacă şi numai dacă impunem următoarele restricţii asupra lui m' (ceea ce justifică calificativul "*restrictiv*" dat acestui protocol de semnătură):

1. m' este de forma m^s cu $m \in G_q$;
2. $\mathcal{U}$ cunoaşte o reprezentare a lui m' în raport cu (g_1, g_2, d) care este multiplu al reprezentării lui m.

Să enunţăm câteva proprietăţi importante ale sistemului Brands:

1. **Securitatea**:
   Din consecinţa anterioară rezultă că identitatea utilizatorului este inclusă în m' ($m' = m^s = I^s d^s$); astfel, dacă utilizatorul foloseşte moneda o singură dată, îi este garantată anonimitatea.
   În schimb, dacă moneda este folosită de două ori, se poate detecta identitatea utilizatorului care trişează, conform Teoremei 7.4.

Securitatea acestui sistem se bazează pe presupunerea de $\mathcal{NP}$ - completitudine a problemei reprezentării în grupuri, a problemei determinării cheii unice Diffie - Hellman şi pe ipoteza că funcţia de dispersie folosită în sistem este criptografică (cu coliziuni tari).

Falsificarea unei monede presupune imitarea semnăturii băncii, ceea ce este calculabil dificil.
Presupunând – totuşi – că $\mathcal{U}$ fasifică o monedă $(m', sig_K(m'))$ (dispunând de o putere de calcul foarte mare), pentru a avea vreo şansă ca moneda să îi fie acceptată în protocolul de plată, el trebuie să cunoască o reprezentare a lui m' în raport cu (g_1, g_2, d) (din Teorema 7.3). Astfel, sarcina falsificatorului devine şi mai dificilă.

2. Anonimitatea utilizatorilor este de asemenea garantată de Teorema 7.3.

3. Sistemul este **fără legături**.
O monedă nu poate fi legată de nici o execuţie specifică a protocolului de extragere, conform Lemei 7.4.

4. Sistemul este **fără urmărire**.
Tranzacţiile utilizatorului nu pot fi urmărite de bancă după protocolul de extragere a monedelor. În momentul în care moneda este depozitată în bancă de către un magazin, banca nu poate deduce din care cont a fost extrasă iniţial.

5. Sistemul este **off-line**.

6. Sistemul asigură protejarea utilizatorilor împotriva acuzării nedrepte a băncii.
Dacă banca acuză un utilizator $\mathcal{U}$ că a folosit o monedă de două ori, ea trebuie să dezvăluie juriului o reprezentare a identităţii lui $\mathcal{U}$, de exemplu (u_1, u_2).

   Dacă $\mathcal{U}$ este onest, reprezentarea (u_1, u_2) coincide cu cea cunoscută de $\mathcal{U}$ cu o probabilitate $1/q$. Deci, cu o probabilitate mare $(1 - \frac{1}{q})$, cele două reprezentări sunt distincte. Astfel, $\mathcal{U}$ se află în posesia a două reprezentări ale lui I în raport cu (g_1, g_2), deci poate calcula o reprezentare a lui 1 în raport cu (g_1, g_2).

   $\mathcal{U}$ prezintă juriului această reprezentare a lui 1, ceea ce constituie proba nevinovăţiei sale ($\mathcal{U}$ nu ar fi putut calcula altfel o reprezentare a lui 1, deoarece aceasta este o problemă $\mathcal{NP}$ - completă).

7.7 Diviziunea monedelor în sistemele de plată electronice

În sistemul de plată descris de Okamoto şi Ohta ([65]) – ineficient din punct de vedere al dimensiunii mesajelor transmise – apare pentru prima oară ideea de a introduce proprietatea de divizibilitate a monedelor electronice.

Ideea folosită este următoarea: Pentru fiecare monedă de valoare d se asociază un arbore binar etichetat astfel: nodul rădăcină (de nivel 0) are valoarea d, cele două noduri de pe nivelul 1 au etichetele $d/2$ şi – în general – nodurile de pe nivelul k sunt etichetate cu $d/2^k$.

Dacă $d = 2^s$, arborele are $s + 1$ nivele, cu frunzele având valoarea minimă 1.

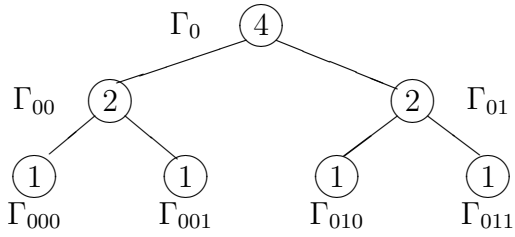
Orice sumă mai mică decât d poate fi cheltuită după următoarele reguli:

1. Odată ce un nod este folosit, toţi succesorii şi predecesorii săi nu vor putea fi folosiţi la o plată viitoare;

2. Nici un nod nu poate fi folosit mai mult decât o singură dată.

Aceste două reguli asigură utilizarrea unui singur nod pentru orice drum de la rădăcină la o frunză, ceea ce face imposibilă folosirea unei valori mai mari decât valoarea rădăcinii d.

Exemplul 7.1. *Să presupunem că Alice dispune de o monedă în valoare de* 4 *unităţi. Se construieşte arborele binar:*



Pentru a realiza o plată de 3 *unităţi, Alice foloseşte* Γ_{00} *şi* Γ_{010}. *Singurul nod care poate fi utilizat la o plată viitoare este* Γ_{011} *în valoare de o unitate.*

Dacă cel puţin una din cele două reguli este încălcată, atunci există un drum de la rădăcină la o frunză, pe care s-au folosit 2 noduri – şi se va putea determina identitatea lui *Alice*.

Mai exact, *Alice* (şi – în general – fiecare utilizator) are asignată o valoare secretă s care relevă identitatea sa, iar fiecare nod i al arborelui are asignată o valoare secretă t_i. Se construieşte dreapta $y = sx + t_i$ pentru fiecare nod i. Când se face o plată utilizând un nod particular n, *Alice* va revela t_i pentru toţi predecesorii lui n.

Magazinul $\mathcal{S}$ va trimite o provocare x_1 la care *Alice* răspunde cu $y_1 = sx_1 + t_n$.

Dacă acelaşi nod n este folosit a doua oară, *Alice* primeşte încă o provocare x_2 la care răspunde cu $y_2 = sx_2 + t_n$, deci ea a utilizat două puncte $(x_1, y_1), (x_2, y_2)$ ale dreptei $y = sx + t_n$, de unde se poate calcula s şi apoi identitatea lui *Alice*.

Dacă se folosesc două noduri n şi m de pe acelaşi drum de la rădăcină la o frunză, presupunem că n este succesor al lui m.

Când se foloseşte nodul n se relevă t_m; apoi – dacă se foloseşte nodul m – *Alice* răspunde la provocarea x_1 a lui $\mathcal{S}$ cu $y_1 = sx_1 + t_m$.

Deci şi în acest caz $\mathcal{S}$ poate calcula s şi identitatea lui *Alice*.

Un dezavantaj al acestei scheme ar fi că se pot realiza legături între plăţi, adică se poate deduce dacă două plăţi provin din aceeaşi monedă şi deci, de la aceeaşi persoană.

În sistemul de plată Brands, pot fi imaginate cel puţin două metode de introducere a monedelor de valori diferite.

- Banca poate să utilizeze câte o cheie publică diferită pentru fiecare tip de monedă. Astfel, dacă în sistem există k monede distincte, banca dispune de k chei publice: $(g_1, h_1), \ldots, (g_k, h_k)$ şi de k chei secrete corespunzătoare: $x_1, \ldots, x_k$ cu $x_i = log_{g_i} h_i$, $\forall i = 1, 2, \ldots k$.

- Se utilizează k generatori $d_1, \ldots, d_k$ (în loc de unul singur d utilizat în sistemul de bază Brands).
  Fiecare generator d_i reprezintă o anumită valoare (de exemplu d_i reprezintă valoarea 2^i). Atunci se poate forma orice monedă de valoare cuprinsă între 1 şi 2^k folosind reprezentarea în binar a valorii monedei. Generatorul d_i apare în reprezentarea monedei dacă bitul $i-1$ din reprezentarea binară este 1.

  De exemplu, pentru o monedă de valoarea $11 = 1011_2$, $m = I \cdot d_4 \cdot d_2 \cdot d_1$, iar pentru o monedă de valoare $18 = 10010_2$, $m = I \cdot d_5 \cdot d_2$.

  Înlocuirea lui $m = I \cdot d$ cu $m = I \cdot d_{i_p} \ldots d_{i_1}$ este singura modificare realizată în protocoalele sistemului de bază Brands.

Protocoalele de plată off-line discutate în acest capitol pot să detecteze identitatea utilizatorilor necinstiţi, fără însă a putea preveni utilizarea unei monede de două ori.
Singurul mod prin care s-ar putea realiza acest lucru este prin încorporarea unei componente hardware (*chip* sau *smart card*) în calculatorul utilizatorului, cu rolul de observator al tuturor plăţilor.

În plus, nici un protocol de plată al utilizatorlui nu poate fi efectuat fără o informaţie secretă deţinută de un observator extern. Acesta autorizează şi participă activ la toate tranzacţiile.

Totuşi, se poate întâmpla ca utilizatorul să reuşească printr-o modalitate oarecare să afle informaţia secretă a observatorului.
În acest caz sistemul trebuie să asigure detectarea utilizatorului care a comis frauda, într-un mod similar protocoalelor prezentate.

Capitolul 8

Protocoale de vot electronic

Guvernele şi organizaţiile democratice au nevoie de anumite mecanisme pentru a permite alegătorilor să îşi exprime opţiunile printr-un proces de votare[1].

În mod tradiţional, alegerile reprezintă – pentru persoanele cu drept de vot – mecanismele oficiale prin care acestea îşi pot exprima opţiunile în mod democratic, în timp ce sondajele constituie una din cele mai bune metode neoficiale de aflare a opţiunilor electoratului. Atât în alegeri cât şi în sondaje intimitatea şi securitatea sunt deziderate obligatorii, dar al căror cost poate duce la creşterea substanţială a procesului de votare.

Mecanismele care asigură securitatea şi intimitatea alegătorilor pot fi scumpe şi/sau consumatoare de timp pentru administratorii acestora şi neconvenabile pentru alegători.

Organizarea de alegeri sigure devine şi mai dificilă atunci când alegătorii sunt distribuiţi pe o mare zona geografică.

Acestea sunt câteva motive pentru care poate fi propusă cu succes modalitatea de vot electronic. Primele idei apar în anii 80, iar ideile dezvoltate în acest interval de timp îl fac din ce în ce mai credibil pentru utilizarea sa într-un mediu informatizat.

8.1 Caracteristici ale unui sistem de vot

Principalele deziderate care trebuie îndeplinite de un sistem de vot electronic sunt:

1. Fiecare persoană poate vota cel mult odată.

2. Voturile sunt anonime (nimeni nu poate afla cui aparţine un anumit vot).

[1]Unul din cele mai vechi protocoale de vot secret stă la baza cuvântului "*a ostraciza*": când cetăţenii vechii Atene credeau că un politician are – prin influenţa pe care a căpătat-o – prea multă putere, ei organizau un proces de votare pentru a decide dacă trebuie să-l exileze pe o perioadă de 10 ani. Fiecare atenian scria pe un ciob de ceramică (în greacă "ostraca") numele persoanei pe care o detesta şi apoi punea acest ciob într-un vas. În final, aceste voturi erau scoase din vas şi numărate, după care un comitet ales de cetăţeni lua decizia.

3. Numărătoarea voturilor trebuie să poată fi verificată de orice sursă externă. În particular, orice alegător poate verifica dacă votul său a fost numărat corect.

4. Nu este posibilă nici o corelare între alegători şi voturi (*receipt - freeness*).

5. Numai alegătorii autorizaţi pot vota (în principiu, pe baza unei liste publice de votanţi).

6. Fiecare alegător este autentificat (pentru a se asigura că el este cel care îşi exercită acest drept).

7. Nu se eliberează nici un fel de dovadă asupra modului cum a votat o persoană (pentru a preveni eventuale vânzări sau constrângeri).

8. Există o limită în timp a procesului de votare.

9. Există o modalitate de a înscrie candidaţii şi de a-i identifica în mod clar pe un buletin de vot.

10. Sistemul se auto-autentifică faţă de alegător (care nu poate modifica însă programele/terminalele de votare).

11. Dacă este permis ca un alegător să îşi schimbe votul înainte de încheierea procesului de votare, acest lucru să fie posibil fără a număra ambele voturi ale alegătorului.

8.1.1 Tipuri de alegeri

Structura protocoalelor de vot depinde în mare măsură de tipul alegerilor; mai exact, depinde de întrebarea pusă alegătorilor şi de răspunsurile posibile.
Se pot distinge următoarele tipuri de alegeri:

- *Votul DA/NU*: Răpunsul alegătorului este dicotomic: DA sau NU. În acest caz, votul poate fi reprezentat pe un singur bit.

- *Votul 1 din L*: Alegătorul are la dispoziţie L variante, din care va alege una. Votul este reprezentat ca un număr întreg din intervalul $[1, L]$.

- *Votul K din L*: Alegătorul selectează K elemente distincte din lista celor L variante propuse. Ordinea elementelor alese nu este importantă. Votul va fi un K-tuplu $(v_1, \ldots, v_K)$.

- *Votul ordonat K din L*: Este varianta anterioară, în care ordinea elementelor din vectorul $(v_1, \ldots, v_K)$ este importantă.

- *Votul 1-L-K*: Alegătorul selectează o listă din cele L variante; apoi alege din această listă K elemente. Votul este un $(K+1)$ - tuplu $(i, v_1, \ldots, v_K)$, unde $v_1, \ldots, v_K$ sunt elemente ale listei i.

- *Votul structurat*: Sunt n liste. Alegătorul le parcurge pe toate, alegând – pentru fiecare $i = 1, 2, \ldots, n$ – maxim k_i elemente distincte din S_i elemente aflate pe lista i. Valorile S_i şi k_i depind de alegerile din listele $1, \ldots, i-1$.
  În acest caz, votul este un tuplu $(v_{11}, \ldots, v_{1k_1}, \ldots, v_{i1}, \ldots, v_{ik_i}, \ldots, v_{nkn})$ unde $\{v_{i1}, \ldots, v_{ik_i}\} \subset S_i$.

Exemplul 8.1. *Un plebiscit (cu întrebări de forma "Sunteţi de acord cu ...") este un vot de tipul DA/NU.*

Alegerea unui conducător dintr-o listă de candidaţi este un exemplu de vot "1 din L".

Pentru alegerile membrilor unui consiliu, alegătorul selectează K persoane din L candidaţi. Candidaţii care au primit cele mai multe voturi vor deveni membrii consiliului.
În acest caz ordinea alegerii candidaţilor nu este importantă.
O variantă de vot este când alegătorul face selecţia într-o anumită ordine, primul selectat primind cele mai multe puncte, iar candidatul K cele mai puţine. Consiliul va fi format din candidaţii care au primit cele mai multe puncte.
Sunt exemple standard de vot "K din L" (ordonat sau nu).

Alegerile parlamentare reprezintă o situaţie când alegătorul alege reprezentanţii de pe lista unui partid. Este un vot de tip 1-L-K, unde L reprezintă numărul de partide politice, iar K – numărul de candidaţi care trebuie aleşi (ordinea selectării lor nu contează).

Votul structurat este o generalizare a votului 1-L-K şi poate fi privit ca o completare a unui formular în care variantele pentru partea următoare depind de alegerile făcute anterior.

Mai sunt şi alte modalităţi de clasificare a voturilor; un exemplu ar fi o clasificare după pondere. În acest caz avem:

- *Votul egal*: Fiecare alegător poate vota o singură dată şi votul său este numărat o singură dată. De exemplu alegerile prezidenţiale sau parlamentare.

- *Votul ponderat*: Votul alegătorului V_i este numărat de w_i ori. Exemplul tipic este cel de la şedinţele acţionarilor, unde fiecare votează cu o pondere dată de numărul acţiunilor pe care le controlează.

8.1.2 Protocolul de vot

Deşi există numeroase protocoale şi sisteme de votare, procedura de bază pentru organizarea de alegeri democratice este standard. Această procedură – indiferent dacă este vorba de o votare clasică sau una electronică – implică în general trei etape:

1. **Iniţializarea**: Autorităţile pregătesc sistemul de vot: anunţă alegerile, formulează întrebările (şi variantele posibile de răspuns), crează listele cu alegătorii eligibili etc. Dacă votul este electronic, generează cheile şi fac publice componentele neprivate ale acestor chei.

2. **Votarea**: Alegătorii îşi exercită dreptul de vot. Comunicaţiile alegătorului cu autorităţile constau în autentificarea sa ca alegător valid, primirea şi completarea unui buletin de vot şi – în final – trimiterea acestui buletin.

3. **Numărarea** Autorităţile colectează buletinele de vot, le deschid, le numără şi apoi anunţă rezultatul alegerilor.

Pentru a avea încredere în rezultatul alegerilor, oamenii trebuie să aibă dovezi că aceste sarcini au fost îndeplinite în mod corect. Există numeroase posibilităţi de corupere a sistemului în timpul îndeplinirii fiecăreia din aceste sarcini. De exemplu:

- Autorităţile electorale pot trişa, permiţând alegătorilor fără drept de vot să se înregistreze la vot, iar alegătorilor înregistraţi să voteze de mai multe ori.
  De asemenea pot pierde sau adăuga voturi suplimentare[2].

- Alegătorii fără drept de vot se pot înregistra (cel mai frecvent folosind numele unei persoane decedate) sau cei cu drept de vot se pot înregistra sub mai multe nume.

- Se poate încerca participarea la vot sub o identitate falsă.

- Cutiile cu voturi, voturile sau maşinile de numărat voturi pot fi compromise.

Atunci când se proiectează un sistem electronic de vot este esenţial să se implementeze modalităţi prin care cele trei etape menţionate mai sus să fie îndeplinite fără a sacrifica intimitatea alegătorilor sau a oferi posibilităţi de fraudă.
În plus, mai trebuie îndeplinite câteva condiţii specifice unui astfel de sistem. Conform cu [51], [61], un sistem de vot electronic trebuie să satisfacă şi următoarele cerinţe:

1. **Anonimitatea votanţilor**: Nu există nici o modalitate prin care poate fi dezvăluită identitatea unei persoane care votează.

2. **Acurateţe**: Oricine poate verifica validitatea voturilor şi se poate asigura că voturile nu au fost schimbate, multiplicate sau eliminate de cineva (inclusiv de autorităţi).

3. **Necoliziune**: Garanţia că toate voturile legale diferă între ele.

[2]Este celebră declaraţia dictatorului rus *I.V.Stalin: Nu este important pe cine votează lumea; important este cine numără voturile.*

4. **Corectitudine de număr**: Suma voturilor repartizate pe candidaţi este egală cu numărul de voturi valide.

5. **Verificabilitate**: Orice alegător poate controla dacă votul său a fost numărat şi repartizat candidatului pe care l-a votat.

6. **Detectarea dublei votări**: Dacă apare un vot dublu, organizatorii pot identifica persoana care a votat de mai multe ori.

8.2 Protocoale bazate pe reţele de permutare

Conceptul de *reţea de permutare.* este introdus de Chaum şi constă din câteva servere interconectate. Fiecare server primeşte o serie de voturi criptate, pe care le amestecă şi scoate o listă de mesaje permutate, astfel încât nu se vor putea face conexiuni între lista primită la intrare şi lista de ieşire.

În aceste sisteme, fiecare alegător îşi generează o cheie (numită adesea *pseudonim*) în colaborare cu autoritatea de vot. Modul de generare se face prin diverse metode (partajare de secrete sau semnături blind), astfel că singurul care ştie cheia este votantul. Identitatea sa nu trebuie dedusă din cheie; deci nimeni nu poate face legătura între alegător, cheie şi votul exprimat (proprietatea *receipt - freeness*) .

8.2.1 Protocolul Chaum

Este primul protocol de vot, propus de Chaum ([12]). Autorităţile sunt N servere, având cheile publice $e_1, \ldots, e_N$, legate într-un centru (panou) de votare unic.

1. Fiecare alegător V_i generează cheia sa publică K_i şi trimite centrului de votare mesajul
$$e_1(e_2(\ldots e_N(K_i)\ldots))$$

2. Serverele decriptează, amestecă cheile K_i de la toţi votanţii şi publică lista acestor chei.

3. Alegătorul V_i verifică dacă cheia sa este pe această listă (dacă nu este, procesul de vot se reia).

4. Fiecare alegător V_i alege opţiunea de vot v_i, după care generează buletinul
$$e_1(e_2(\ldots e_N(K_i \| K_i^{-1}(v_i))\ldots))$$
pe care îl trimite la centrul de votare.

5 Serverele decriptează, elementele $K_i \| K_i^{-1}$ sunt interclasate cu lista cheilor şi se află voturile v_i.

Protocolul verifică multe din proprietăţile unui sistem de vot, dar prezintă şi numeroase dezavantaje; cel mai mare din ele este cel de reluare a sistemului de vot la simpla eşuare a unui votant. Dacă între timp o parte din voturi au fost deja publicate, procesul de alegeri este compromis.
Protocolul Chaum este reluat şi îmbunătăţit în 1993 de C. Park, K. Itoh şi K. Kurosawa.

8.2.2 Protocolul Merritt

Michael Merritt construieşte un protocol destul de asemănător cu cel al lui Chaum, a cărui principală calitate este aceea că nu foloseşte nici o autoritate. Similar protocolului precedent, se bazează pe aplicări succesive de criptări şi/sau semnături digitale de mesaje. Anonimitatea se obţine prin aplicarea de permutări în diverse faze.

Sunt N alegători, fiecare alegător V_i având cheia publică de criptare e_i şi cheia privată d_i, precum şi o semnătură electronică sig_i. Protocolul este următorul:

1. Fiecare alegător V_i ataşează un număr r_i (generat aleator) la votul său v_i.
2. Criptează perechea (v_i, r_i) cu cheile publice ale tuturor alegătorilor în ordinea $1, \ldots, N$, obţinând secvenţa
$$e_N(e_{N-1}(\ldots(e_1(v_i, r_i))\ldots)$$
3. Repetă pasul anterior, adăugând după fiecare criptare câte un număr aleator R_j distinct (pe care-l trimite lui V_j). Se obţine
$$e_N(R_N, e_{N-1}(\ldots e_2(R_2, e_1(R_1, e_N(e_{N-1}(\ldots(e_1(v_i, r_i))\ldots)$$
4. Toate aceste voturi sunt strânse de alegătorul V_N.
În continuare, fiecare alegător V_i:
   (a) Decriptează mesajul cu cheia sa secretă d_i, scoate numărul R_i şi se asigură că este pe lista numerelor aleatoare primite.
   (b) Amestecă cele N voturi şi le trimite lui V_{i-1} (V_1 le trimite lui V_N).

   După efectuarea acestui pas, alegătorul V_N dispune de N mesaje de forma
$$e_N(e_{N-1}(\ldots(e_1(v_i, r_i))\ldots).$$
   pe care le semnează cu semnătura sa sig_N.

5 Procedeul se reia, fiecare alegător V_i:

(a) Verifică validitatea semnăturii alegătorului V_{i+1};

(b) Decriptează mesajele primite;

(c) Aplică propria sa semnătură sig_i şi trimite lui V_{i-1} cele N mesaje de forma

$$sig_i(e_i(\ldots(e_1(v_i, r+i)\ldots).$$

6 Toţi alegătorii confirmă semnătura alegătorului V_1. Voturile sunt numărate în comun, fiecare alegător putându-se convinge de existenţa votului său, datorită numărului aleator ataşat.

În timpul procesului de votare numărul de voturi este constant, deci pierderea sau adăugarea unui vot este uşor de detectat. Amestecul voturilor asigură anonimitatea.
De asemenea, voturile nu pot fi înlocuite; o astfel de încercare pe parcursul primei runde este depistată prin numărul aleator introdus incorect. Dacă alegătorul V_i înlocuieşte votul alegătorului V_j ($j > i$), atunci alegătorul V_j va detecta acest lucru la începutul celei de-a doua runde de decriptări (pasul 5).
O încercare de înlocuire pe parcursul celei de-a doua runde este depistată la decriptarea finală, când fiecare alegător îşi verifică propriul său număr r_i.

Un defect major al acestui protocol constă în implementarea dificilă, datorată numărului mare de calcule, dependent de numărul N de votanţi.

8.2.3 Protocol cu autoritate centrală

Pentru micşorarea volumului de calcule din protocolul Merritt se poate introduce o nouă entitate care se ocupă cu înregistrarea alegătorilor; o vom numi *autoritate centrală* (AC). Se presupune că prin carta de alegător, fiecare persoană V_i dispune de două chei e_i (publică) şi d_i (secretă). Protocolul de vot este prezentat pe pagina următoare.

Există şi în acest protocol câteva neajunsuri. Primul – şi cel mai important – este acela că o autoritate centrală reprezintă un punct de corupţie asupra căruia nu există control. AC poate falsifica voturi în numele alegătorilor care se abţin, sau poate pierde voturi valide (nici un alegător nu poate demonstra că a trimis într-adevar un vot).
În plus, implementarea sa rămâne destul de complexă.

O primă idee de îmbunătăţire a sistemului a constat în introducerea mai multor autorităţi centrale, specializate pe anumite operaţii. De exemplu, se pot introduce două autorităţi: una care se ocupă de legitimarea alegătorilor (să îi spunem AL - agenţie de legitimitate), alta care se ocupă de numărarea efectivă a voturilor (AT - agenţie de tabulare).

1. AC publică o listă cu toţi alegătorii înregistraţi.
2. Fiecare alegător primeşte de la AC un ID (printr-un protocol de dezvăluire parţială a secretelor, similar celor de la protocoalele electronice de plată).
3. Fiecare alegător V_i trimite spre AC perechea $(ID, e_i(ID, v_i))$.
4. AC publică $e_i(ID, v_i)$ pentru toţi alegătorii de pe lista de la punctul (2).
5. Fiecare alegător V_i trimite anonim spre AC perechea (ID, d_i).
6. AC asociază mesajele după ID, le decriptează, verifică autenticitatea voturilor şi publică perechile (ID, v_i) pentru toţi participanţii la vot.

Un vot valid trebuie să treacă prin ambele agenţii pentru validare. Prima recunoaşte dreptul alegătorului de a vota (fără a vedea conţinutul votului), eliberându-i un buletin. A doua agenţie primeşte votul, împreună cu buletinul de validare. O variantă de astfel de protocol (în ipoteza că cele două agenţii nu se aliază pentru falsificarea votării) este:

1. Fiecare alegător (după demonstrarea identităţii sale) solicită de la AL un număr de autentificare.
2. AL generează aleator numere de autentificare şi le distribuie.
3. AL trimite spre AT lista tuturor numerelor de autentificare.
4. Fiecare votant alege aleator un ID (număr de validare) şi trimite spre AT un triplet format din numărul de autentificare, ID şi votul său.
5. AT verifică numărul de autentificare şi – dacă este pe listă – îl bifează şi publică votul împreună cu nunmărul de validare.

8.3 Protocoale bazate pe semnături blind

Semnăturile blind sunt folosite frecvent ca metode de autentificare a mesajelor. În general, având un mesaj α, *Alice* îl "ascunde" folosind o valoare aleatoare r; fie $x = f(\alpha, r)$. Autoritatea semnează x oferind $sig_A(x)$; din aceasta, *Alice* extrage $sig_A(\alpha)$. Pentru detalii, a se revedea secţiunea 3.10.

Protocoalele de votare prezentate în această secţiune funcţionează de obicei cu două autorităţi (distincte sau incluse în aceeaşi autoritate centrală): un *Administrator* – care

semnează blind mesajul trimis de alegător, şi un *Colector* – însărcinat cu strângerea, publicarea şi numărarea voturilor.

Această abordare este mai sigură decât variantele anterioare, dar conţine şi unele falii de securitate, care trebuie rezolvate punctual. Cele mai importante slăbiciuni sunt:

- Lipsa imparţialităţii: colectorii cunosc rezultatele intermediare înainte de faza de numărare.
- Netratarea coliziunilor: există o şansă ca doi alegători să trimită acelaşi mesaj la *Administrator*; atunci unul din voturi va fi exclus, ca fiind vot dublu.
- O autoritate coruptă (*Administratorul*) se poate da drept alegător – în locul votanţilor absenţi – şi poate opera în locul lor; sau poate transmite în secret mai multe mesaje anumitor alegători.
- Dacă votul unui alegător nu a fost luat în considerare, acesta nu poate contesta acest lucru fără a-şi dezvălui votul.

Cele două protocoale prezentate în continuare încearcă să rezolve o parte din aceste falii de securitate.

8.3.1 Protocolul *FOO*

Protocolul Fujioka, Okamoto şi Ohta prezentat în 1992 ([33]) este primul sistem de vot din această clasă care asigură confidenţialitatea şi imparţialitatea.
Vom detalia acest protocol, construit pe patru etape:

- **Etapa de iniţializare**:
  - *Administratorul* defineşte funcţia de semnătură (sig_A, ver_A).
    ver_A este făcută publică.
  - *Colectorul* defineşte un număr prim p pentru care problema logaritmului discret în Z_p este dificilă, şi un generator $g \in Z_p$. Apoi alege aleator $a \in Z_{p-1}$ şi calculează $\alpha = g^a \ (mod\ p)$.
    De asemenea, detaliază funcţia de ascundere f şi de dezvăluire g, necesare protocolului de semnătură blind.
    Elementele (p, g, α) precum şi (f, g) sunt publice.
  - Pentru fiecare alegător V_i se definesc identificatorul ID_i şi protocolul de semnătură (sig_i, ver_i). De asemenea, V_i generează aleator $r_i, k_i \in Z_{p-1}$.
    ID_i şi ver_i sunt publice, celelalte sunt secrete.

- **Etapa de înregistrare**:

  - Alegătorul V_i pregăteşte buletinul de vot astfel:
    1. Selectează votul v_i şi crează buletinul de vot $x_i = g^{k_i}\alpha^{v_i} \ (mod\ p)$.
    2. Ascunde acest buletin în $e_i = f(x_i, r_i)$.
    3. Semnează mesajul: $s_i = sig_i(e_i)$ şi trimite *Administratorului* tripletul (ID_i, e_i, s_i).
  - *Administratorul* primeşte tripletul şi:
    1. Verifică dacă V_i are drept de vot şi dacă $ver_i(e_i, s_i) = True$.
    2. Dacă testul este trecut cu succes, generează $d_i = sig_A(e_i)$ pe care-l trimite lui V_i.

  La sfârşitul etapei de înregistrare, *Administratorul* anunţă numărul de alegători care au primit semnătura sa şi publică lista tripletelor (ID_i, e_i, s_i).

- **Etapa de votare**:

  - Alegătorul V_i primeşte d_i şi
    1. Obţine – cu funcţia de dezvăluire – semnătura $y_i = g(d_i, r_i) = sig_A(x_i)$.
    2. Verifică $ver_A(x_i, y_i) = True$. Dacă egalitatea nu este adevărată, solicită întreruperea alegerilor, arătând că perechea (x_i, y_i) este invalidă.
    3. Trimite *Colectorului* printr-un canal anonim perechea (x_i, y_i).
  - *Colectorul* verifică şi el $ver_A(x_i, y_i) = True$. Dacă verificarea are succes, introduce (n, x_i, y_i) într-o listă, ca al n-lea element.
    După terminarea votării, *Colectorul* face publică această listă.

  Prin *canal anonim* se înţelege un canal de comunicaţie în care identitatea expeditorului nu se poate determina.

- **Etapa de numărare**:

  - (**Deschiderea**) Fiecare alegător V_i efectuează următoarele operaţii:
    1. Verifică dacă numărul de buletine de vot este egal cu numărul de alegători şi dacă buletinul său de vot este prezent în lista de buletine. Dacă nu se află, V_i dezvăluie (x_i, y_i).
    2. Trimite *Colectorului* printr-un canal anonim perechea (n, k_i).

– (**Numărarea**) *Colectorul*:

1. Din egalitatea $x_i = g^{k_i}\alpha^{v_i}$ află v_i (ca element dintr-o mulţime finită de cardinal mic).
2. Adaugă k_i, v_i la poziţia n din listă şi verifică dacă votul v_i este valid.
3. Numără voturile şi publică rezultatul alegerilor.

Proprietăţi ale protocolului *FOO*:

- Doar alegătorii cu drept de vot au tripletele (ID, e, s) publicate pe lista *Administratorului*. Tripletele şi voturile (x_i, y_i) invalide sunt detectate. Un alegător nu poate vota decât o singură dată.

- Confidenţialitatea alegătorului este păstrată chiar şi în cazul unei alianţe între *Administrator* şi *Colector*. Protocolul de semnătură blind şi trimiterea buletinului de vot printr-un canal anonim asigură această confidenţialitate.

- Orice alegător poate verifica (individual) dacă buletinul său de vot (x, y) se află pe lista publicată de *Colector* şi dacă (k_i, v_i) au fost adăugate în final la listă.
  Când un alegător cere întreruperea alegarilor, el nu trebuie să-şi dezvăluie votul său ci doar perechea (x, y). Alegătorii care au făcut reclamaţie şi au prezentat perechi (x, y) valide se pot înregistra a doua oară pentru a obţine o nouă astfel de pereche. Totuşi dacă această reclamaţie are loc în faza de numărare şi alegerile continuă, proprietatea de confidenţialitate este violată pentru alegătorii reclamanţi.

- Protocolul de vot este imparţial – numărarea voturilor nu afectează votarea, ea fiind făcută într-o etapă ulterioară.

Sunt totuşi două proprietăţi – importante pentru un sistem de vot – care nu sunt îndeplinite de protocolul *FOO*:

1. **Verificabilitatea universală**: Dacă anumiţi alegători se abţin de la vot după etapa de înregistrare, *Administratorul* poate să adauge voturile lor, fără a putea fi depistat.

2. **Receipt - Freeness**: Orice persoană care reuşeşte să afle perechea (x, y) pentru un anumit alegător, poate afla uşor votul său publicat de *Colector* la sfârşit.

O variantă îmbunătăţită a protocolului *FOO* – care rezolvă şi aceste condiţii – este prezentată de Tatsuaki Okamoto în 1997 ([64]).

8.3.2 Protocolul Radwin

Acest protocol (propus în 1995 de Michael Radwin) are multe puncte comune cu protocolul de semnătură Lamport şi cu protocolul de plată Chaum - Fiat - Naor. El este format dintr-o singură autoritate AC (îndeplinind atât rolul de *Administrator* cât şi cel de *Colector*) şi necesită existenţa unui canal anonim securizat care suportă replay-uri (destinatarul mesajului anonim poate trimite un replay expeditorului).

Fie f, g funcţii de dipersie criptografică cu două argumente, considerate publice.

- **Etapa de iniţializare**:
  AC crează şi publică cheia sa publică RSA (n, e) şi un parametru de securitate t.

- **Etapa de înregistrare**:
  Alegătorul V – cu identificatorul ID – îşi construieşte pseudonimul P conform protocolului următor:

1. Alegătorul V:
   (a) Generează aleator $a_k, c_k, d_k, r_k \in Z_n \quad (1 \leq k \leq 2t)$;
   (b) Calculează $x_k = g(a_k, c_k),\ y_k = g(a_k \oplus ID, d_k)$ şi $B_k = r_k^e \cdot f(x_k, y_k)$;
   (c) Trimite autorităţii AC vectorul $B = (B_1, \ldots, B_{2t})$.
2. AC selectează aleator o mulţime $R \subset \{1, 2, \ldots, 2t\}$ de t valori, pe care o trimite lui V.
3. V trimite lui AC valorile a_k, c_k, d_k, r_k cu $k \in R$.
4. Autoritatea AC:
   (a) Calculează (pentru fiecare $k \in R$) valorile x_k, y_k şi verifică egalităţile
   $$B_k = r_k^e \cdot f(x_k, y_k) \pmod n$$
   Dacă relaţiile sunt verificate pentru toate valorile din R, atunci valorile din B rămase neverificate sunt considerate corecte.
   Dacă nu, autoritatea respinge înregistrarea lui V.
   (b) Calculează $S_k = B_k^d \pmod n$ pentru valorile $k \notin R$;
   (c) Trimite lui V valoarea $S = \prod_{k \notin R} S_k$.
5. V îşi calculează pseudonimul $P = \prod_{k \notin R} f(x_k, y_k)$ şi semnătura

$$SP = \frac{S}{\prod_{k \notin R} r_k} = \prod_{k \notin R} f(x_k, y_k)^d \pmod n$$

De remarcat că valorile calculate la pasul $4(b)$ sunt

$$S_k = B_k^d = [r_k^e \cdot f(x_k, y_k)]^d = r_k \cdot f(x_k, y_k)^d \pmod n,$$

deci

$$S = \prod_{k \notin R} \left[r_k \cdot f(x_k, y_k)^d \right]$$

Deoarece elementele a_k, c_k, d_k cu $k \in R$ nu mai sunt de folos în continuare, vom considera – pentru simplificarea notaţiei – că R a fost mulţimea $\{t+1, \ldots, 2t\}$.

- **Etapa de votare**:
Protocolul de votare este următorul:

1. Alegătorul V:
   (a) Alege votul v şi crează buletinul de vot (v, P, SP);
   (b) Trimite spre AC – printr-un canal anonim – mesajul
   $$BV = (v, P, SP)^e \pmod n$$
2. Autoritatea AC:
   (a) Decriptează mesajul primit şi verifică congruenţa $P \equiv SP^e \pmod n$.
   (b) Dacă testul se verifică, generează aleator un vector binar $Z = (z_1, \ldots, z_t)$, pe care-l trimite – ca replay – lui V.
3. V răspunde cu vectorul de triplete $S = (s_1, \ldots, s_t)$, unde – pentru $k = 1, 2, \ldots, t$:
   - Dacă $z_k = 0$ atunci $s_k = (a_k, c_k, y_k)$;
   - Dacă $z_k = 1$, atunci $s_k = (x_k, a_k \oplus ID, d_k)$
4. Pentru fiecare triplet $s_k = (s_{k_1}, s_{k_2}, s_{k_3})$, autoritatea AC
   (a) Calculează p_k astfel:
   - Dacă $z_k = 0$ atunci $p_k = f(g(s_{k_1}, s_{k_2}), s_{k_3})$;
   - Dacă $z_k = 1$ atunci $p_k = f(s_{k_1}, g(s_{k_2}, s_{k_3}))$.
   (b) Verifică egalitatea $P = \prod_{k=1}^{t} p_k \pmod p$.
   (c) Dacă egalitatea este verificată, nu există nici un vot primit anterior de la alegătorul cu pseudonimul P şi v este un vot valid, atunci AC numără acest vot.

De remarcat că dacă P a fost utilizat de două ori, atunci AC poate determina – cu probabilitate mare – identitatea lui V în felul următor:
Când V trimite un pseudonim P pentru prima (a doua) oară, el este provocat cu

un vector binar aleator Z_1 (Z_2) să dezvăluie parţial structura lui P. Z_1 şi Z_2 diferă – cu probabilitate foarte mare – cel puţin printr-un indice k. Tripletele s_k din Z_1 şi Z_2 conţin a_k şi $a_k \oplus ID$, de unde se poate afla imediat ID-ul lui V.

- **Etapa de numărare**:
  AC numără voturile valide şi publică suma finală. Sunt posibile două variante ale acestei etape:
  - *Varianta fără listă*: Este făcută publică numai suma finală a voturilor.
  - *Varianta cu listă*: AC publică buletinele de vot primite, pseudonimele P, SP, provocările Z cu răspunsurile S şi voturile corespondente v.

Protocolul de votare Radwin verifică aceleaşi proprietăţi ca şi FOO. De remarcat că nici acest protocol nu asigură verificabilitatea universală (AC poate adăuga voturi în locul votanţilor absenţi sau poate oferi unui alegător posibilitatea de a-şi genera mai multe pseudonime).

Proprietatea *Receipt-freeness* poate fi îndeplinită doar în varianta de numărare fără listă – dacă V trimite buletinul de vot printr-un canal anonim. Altfel (dacă se operează pe un canal neanonim sau în varianta de numărare cu listă), oricine poate face legătura între votantul V şi votul său v.

8.4 Protocoale bazate pe criptări homomorfe

Conceptul este definit formal în [23], iar primul protocol de acest gen aparţine lui Benaloh ([4]). Ideile sale sunt prezente în majoritatea protocoalelor actuale de vot.

O funcţie $f : A \longrightarrow B$ este *homomorfă* dacă

$$f(x) \cdot f(y) = f(x + y), \quad \forall x, y \in A$$

Într-o schemă de vot homomorfă, fiecare alegător V îşi criptează votul folosind o funcţie de criptare homomorfă, cu cheie publică. Ulterior, V trebuie să demonstreze că votul său criptat este criptarea unui vot valid (în general, schemele de vot din această clasă se diferenţiază la acest pas).

Voturile criptate sunt însumate folosind proprietatea de homomorfism a funcţiei de criptare. În final, o serie de observatori neutri cooperează pentru a decripta rezultatul final (cheia secretă a sistemului de criptare fiind partajată între aceşti observatori, utilizaând o schemă de partajare a secretelor).

Protocoalele de vot bazate pe criptări homomorfe sunt eficiente şi pot fi verificabile: multe operaţii cu voturile criptate pot fi operate public, chiar în timpul procesului de votare, fără a interacţiona cu autorităţile de vot.

8.4.1 Protocolul de vot Benaloh

În forma sa inițială, este o schemă de votare de tip *DA/NU*.
În el sunt implicate N autorități $A_1, \ldots, A_N$ din care cel puțin t sunt de încredere.

- **Etapa de inițializare**:
  Fiecare autoritate A_j își generează cheia de criptare (publică) și de decriptare.
  Vom nota cu $e_j(m, k)$ criptarea unui mesaj m cu cheia publică a autorității A_j și parametrul de criptare k (autorul a utilizat sistemul de criptare ElGamal).

  Este definită de asemenea o constantă de securitate t.

- **Etapa de votare**: Alegătorul V trimite valoarea 0 sau 1 după următorul algoritm:

1. V generează perechea (v_1, v_2) ca o permutare a lui $(0, 1)$.
2. Pentru fiecare $i = 1, 2$ partajează secretul v_i în $s_1(v_i), \ldots, s_N(v_i)$ utilizând protocolul de partajare a secretelor (t, N) al lui Shamir.
3. Calculează
$h_i = (h_{i_1}, \ldots, h_{i_N}),\ \ i = 1, 2 \quad$ unde $\quad h_{i_j} = e_j(s_j(v_i), k_{ij}), \quad j = 1, 2, \ldots, N.$
4. Publică $h = (h_1, h_2)$ și dovedește autorităților că h este corect construit.
5. Alege ca vot dorit h_1 sau h_2.

V dovedește corectitudinea votului său urmând un protocol de tip provocare/răspuns:

1. V creează T perechi de voturi criptate $h^1, \ldots, h^T$:
$$h^r_{ij} = e_j(s^r_j(v^r_i), k^r_{ij}), \quad 1 \le r \le T,\ \ 1 \le j \le N,\ \ i = 1, 2$$
pe care le trimite autorităților.
2. Autoritățile generează aleator T biți $c_1, \ldots, c_T$ pe care îi trimit lui V.
3. - Pentru fiecare $c_r = 0$, V transmite valorile
$$v^r_i,\ s^r_j(v^r_i),\ k^r_{ij} \quad (1 \le j \le N,\ i = 1, 2),$$
dezvăluind astfel cum a fost construită perechea $h^r = (h^r_1, h^r_2)$.
- Pentru fiecare $c_r = 1$, V dezvăluie permutarea care duce perechea (v_1, v_2) în (v^r_1, v^r_2): presupunem că $v_1 = v^r_1,\ v_2 = v^r_2$.
Atunci $h_{ij} = e_j(s_j(v_i), k_{ij})$ și $h^r_{ij} = e_j(s^r_j(v^r_i), k^r_{ij})$.
Din proprietatea de homomorfism rezultă faptul că $h/h^r = e_j(s_j(v_i) - s^r_j(v_i), k_{ij}/k^r_{ij})$ este criptarea votului $(0, 0)$.
V transmite autorităților valorile $s_j(v_i) - s^r_j(v_i)$ și k_{ij}/k^r_{ij}.

4 Autorităţile verifică dacă răspunsul se potriveşte cu h:
- Pentru toţi biţii $c_r = 0$ verifică dacă h^r este creat corect;
- Pentru toţi biţii $c_r = 1$ verifică dacă h_{ij}/h_{ij}^r este criptarea votului $(0, 0)$.

- **Etapa de numărare**:
  Fie $h_i = (h_{i_1}, \ldots, h_{i_N})$ votul alegătorului V. Autoritatea A_j calculează

$$\prod_i h_{ij} = \prod_i e_j(s_j(v_i)) = e_j\left(\sum_i s_j(v_i)\right)$$

  de unde – prin decriptare – determină suma părţilor sale: $S_j = \sum_i s_j(v_i)$.

  Valorile S_j (eventual împreună cu o dovadă de corectitudine a decriptării) sunt făcute publice.

  Fie A mulţimea de t autorităţi care au avut succes în decriptarea părţilor lor S_j. Suma finală a voturilor poate fi calculată acum de oricine astfel:

$$S = \sum_{j\in A} S_j\lambda_j = \sum_{j\in A}\left(\sum_i s_j(v_i)\right)\lambda_j = \sum_i\sum_{j\in A} s_j(v_i)\lambda_j = \sum_i v_i$$

  unde λ_j sunt coeficienţii Lagrance din schema Shamir de partajare a secretelor (Capitolul 4, secţiunea 4.1.2).

Protocolul Benaloh are proprietăţile principale (eligibilitate, confidenţialitate, verificabilitate universală). Nu verifică *receipt - freeness*, deoarece alegătorul V trebuie – pentru a dovedi votul său – să prezinte parametrii aleatori k_{ij} utilizaţi în criptare.

Votul Benaloh poate fi extins la diverse moduri de vot, cum ar fi:

- *Votul* 1 *din* L: Permutarea $v_1, \ldots, v_L$ a celor L posibilităţi este criptată în $h = (h_1, \ldots, h_L)$. Cele L posibilităţi pot fi reprezentate ca $1, M, M^2, \ldots, M^L$ (unde M este numărul alegătorilor).
  În acest fel se poate determina uşor rezultatul alegerilor din suma tuturor voturilor.

- *Votul ponderat*: Dacă votul alegătorului V trebuie numărat de n ori, L-tuplul său va fi $(n \cdot v_1, \ldots, n \cdot v_L)$.

8.4.2 Protocolul de vot Schoenmakers

Este similar cu protocolul precedent: alegătorul partajează votul său între N autorităţi, utilizând un protocol special. Pentru determinarea rezultatului final se foloseşte proprietatea de homomorfism a funcţiei de partajare.

Prezentăm o variantă simplificată a protocolului, propusă de Cramer, Franklin, Schoemakers şi Yung ([23]) şi completată ulterior de Schoemakers în [71]:

- **Etapa de iniţializare**:
  Cele N autorităţi publică numerele prime p, q cu $q|(p-1)$, $m < q/2$, elementele $\alpha, h \in Z_p$ de ordin q, precum şi un parametru de securitate $k \in [2, N]$.

- **Etapa preliminară de votare**:

1. Alegătorul V_j $(1 \leq j \leq m)$ alege o valoare $b_j \in \{-1, 1\}$ şi un număr aleator $\delta_j \in Z_q^*$.
2. Calculează $B_j = \alpha^{\delta_j} h^{b_j} \pmod p$.
3. Alege polinoamele $G_j(X),\ H_j(X) \in Z_q^*[X]$:
$G_j(X) = \delta_j + \delta_{j,1}X^1 + \ldots + \delta_{j,k-1}X^{k-1},\ H_j(X) = b_j + \beta_{j,1}X^1 + \ldots + \beta_{j,k-1}X^{k-1}$

4 Face public B_j împreună cu $proof(B_j)$ ($proof(B_j)$ este un protocol de tip provocare/răspuns prin care V_j arată că B_j corespunde unui element $b_j \in \{-1, 1\}$).
Orice autoritate poate verifica corectitudinea lui B_j folosind $proof(B_j)$.

5 Calculează
$$B_{j,t} = \alpha^{\delta_{j,t}} h^{\beta_{j,t}} \pmod p, \quad t = 1, \ldots, k-1$$
pe care le face publice.

6 Calculează $a_{i,j} = G_j(i),\ b_{i,j} = H_j(i)$ şi trimite – printr-un canal securizat – perechea $(a_{i,j}, b_{i,j})$ autorităţii A_i, $(1 \leq i \leq N)$.

7 Autoritatea A_i poate verifica corectitudinea informaţiei primite de la alegătorul V_j verificând dacă este îndeplinită congruenţa
$$\alpha^{a_{i,j}} h^{b_{i,j}} \pmod p \equiv B_j \prod_{t=1}^{k-1} B_{j,t}^{i^t} \pmod p$$

- **Etapa de votare**:

Alegătorul V_i alege votul său $v_j \in \{-1, 1\}$ şi trimite elementul s_j determinat de egalitatea $v_j = s_j \cdot b_j$.

- **Etapa de numărare**:

1. Fiecare autoritate A_i face publică perechea (S_i, T_i), unde
$$S_i = \sum_{j=1}^{m} b_{i,j} s_j \pmod p, \qquad T_i = \sum_{j=1}^{m} a_{i,j} s_j \pmod p$$
2. Oricine poate certifica această informaţie verificând egalitatea
$$\alpha^{T_i} h^{S_i} \pmod p = \prod_{j=1}^{m} \left(B_j \prod_{t=1}^{k-1} B_{j,t}^{i^t} \right)^{s_j} \pmod p$$
3. Autoritatea centrală (sau oricine este interesat) poate calcula în final suma
$$S = \sum_{i \in A} \left(S_i \prod_{j \in A \setminus \{i\}} \frac{j}{j-i} \right) \pmod q$$
unde $A \subseteq \{1, \ldots, N\}$ este o mulţime arbitrară de k elemente pentru care informaţia $(S_i, T_i),\ i \in A$ este consistentă.

Şi această schemă de vot verifică principalele cerinţe, mai puţin cea de *receipt-freness*.

Sunt multe variante de construcţie a unui protocol $proof(B)$ care să demonstreze că $B = \alpha^\delta h^b$ este construit cu un $b \in \{-1, 1\}$. (α şi h sunt publice)
Un exemplu de astfel de protocol este următorul:

1. V alege aleator $c_1 \in Z_q^*$ şi calculează
$$a_1 = \left(\alpha^\delta h\right)^{c_1}, \quad a_2 = \left(\alpha^\delta h^{-1}\right)^{c_1}$$
(calcule modulo p). Face public tripletul (c_1, a_1, a_2).
2. O autoritate A care doreşte să verifice, testează egalitatea
$$a_1 \cdot a_2^{-1} = h^{2c_1} \pmod p.$$
Dacă este verificată, trimite lui V o valoare $c \in Z_q^*$ generată aleator.
3. V calculează $c_2 = c \cdot c_1^{-1} \pmod q$ şi $x = \left(\alpha^\delta\right)^c$. Trimite lui A perechea (c_2, x).
4. A verifică egalităţile
$$c = c_1 \cdot c_2 \quad \text{şi} \quad (a_1 \cdot a_2)^{c_2} = x$$
Dacă sunt îndeplinite, A ştie că pentru construcţia lui $B = \alpha^\delta h^b$ s-a folosit o valoare b din mulţimea $\{-1, 1\}$.

Anexa 1

Elemente de algebra curbelor eliptice

1.1 Algebra curbelor eliptice

Fie K_0 un corp finit de caracteristică $p > 3$ (în general se consideră $K_0 = Z_p$).

Vom numi curbă eliptică peste K_0 congruenţa

$$y^2 \equiv x^3 + ax + b \ \ (mod\ p)$$

unde $a, b \in K_0$. Descriminantul ei este $\Delta = -16(4a^3 + 27b^2)$.

Dacă K este o extensie oarecare a lui K_0, atunci mulţimea punctelor curbei E pe K este

$$E(K) = \{(x, y) \in K \times K \mid y^2 \equiv x^3 + ax + b \ \ (mod\ p)\} \bigcup \{\mathcal{O}\}$$

unde $\mathcal{O}$ este "*punctul de la infinit*".

O curbă eliptică E se poate structura ca un grup abelian finit. Legea de compoziţie (notată aditiv) este definită astfel:

Fie $P, Q \in E(K), \quad P = (x_1, y_1),\ Q = (x_2, y_2)$.

Dacă $x_2 = x_1,\ y_2 = -y_1$, atunci $P + Q = \mathcal{O}$; altfel, $P + Q = (x_3, y_3)$ unde

$$x_3 = \lambda^2 - x_1 - x_2, \quad y_3 = \lambda(x_1 - x_3) - y_1,$$

iar

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

Se mai defineşte $P + \mathcal{O} = \mathcal{O} + P = P,\ \forall P \in E(K)$.

Verificarea proprietăţilor de grup este banală[1]. Elementul neutru este $\mathcal{O}$.

De remarcat că inversa lui (x, y) (notată $-(x, y)$) este $(x, -y)$.

[1]Pentru detalii, a se vedea şi [2].

Exemplul 1.1. *Fie E curba eliptică $y^2 = x^3 + x + 5$ peste Z_{19}. Să calculăm la început punctele lui $E(Z_{19})$. Aceasta se face astfel: $\forall x \in Z_{19}$ se calculează $z = x^3+x+5 \ (mod\ 19)$; apoi se testează dacă z este rest pătratic.*

Rezultatele sunt strânse în tabelele următoare (toate calculele se realizează modulo 19):

a	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
a^2	0	1	4	9	16	6	17	11	7	5	5	7	11	17	6	16	9	4	1

x	x^3+x+5	y	x	x^3+x+5	y	x	x^3+x+5	y
0	5	9, 10	1	7	8, 11	2	15	–
3	16	4, 15	4	16	4, 15	5	2	–
6	18	–	7	13	–	8	12	–
9	2	–	10	8	–	11	17	6, 13
12	16	4, 15	13	11	7, 12	14	8	–
15	13	–	16	13	–	17	14	–
18	3	–						

Curba eliptică E admite deci 15 puncte; cum ordinul grupului $E(Z_{19})$ nu este număr prim, grupul nu este ciclic. Vom alege un element primitiv drept generator. Fie acesta $\alpha = (0, 9)$. Calculăm "puterile" lui α (de fapt multiplii, grupul fiind aditiv).
Pentru 2α se calculează întâi (modulo 19):

$$\lambda = (3 \cdot 0^2 + 1)(2 \cdot 9)^{-1} = 1 \cdot 18^{-1} = 18.$$

Acum se pot determina
$x_3 = 18^2 - 0 - 0 = 361 \equiv 1 \ (mod\ 19), \quad y_3 = 18 \cdot (0 - 1) - 9 = -27 \equiv 11 \ (mod\ 19)$,
deci $2\alpha = (1, 11)$.
Multiplul următor este $3\alpha = 2\alpha + \alpha = (1, 11) + (0, 9)$. Avem:

$$\lambda = (9 - 11) \cdot (0 - 1)^{-1} = 2, \text{ deci}$$

$x_3 = 2^2 - 1 - 0 = 3, \quad y_3 = 2 \cdot (1 - 3) - 11 = -15 \equiv 4 \ (mod\ 19)$,
de unde rezultă $3\alpha = (3, 4)$.
În mod similar se obţin toate punctele curbei eliptice E:

$\alpha = (0, 9)$	$2\alpha = (1, 11)$	$3\alpha = (3, 4)$	$4\alpha = (4, 4)$	$5\alpha = (13, 12)$
$6\alpha = (11, 6)$	$7\alpha = (12, 15)$	$8\alpha = (12, 4)$	$9\alpha = (11, 13)$	$10\alpha = (13, 7)$
$11\alpha = (4, 15)$	$12\alpha = (3, 15)$	$13\alpha = (1, 8)$	$14\alpha = (0, 10)$	$15\alpha = \mathcal{O}$

De remarcat că – de exemplu – $(3, 4)$ nu este element primitiv, având ordinul 5.

Definiţia 1.1. *Considerăm o curbă eliptică E peste Z_q, $m = card(E(F_q))$ şi n un număr întreg cu proprietatea $n|m$. Fie $k > 1$ cel mai mic număr întreg cu proprietatea $n|(q^k - 1)$. k se numeşte "gradul de acoperire al curbei E în raport cu n".*
Dacă $n = m$, atunci k este "gradul de acoperire al curbei E" (în engleză "embedding degree").

Cu ajutorul gradului de acoperire k, putem considera $GF(q^k)$ ca fiind o extensie a lui Z_q, iar $E(Z_q)$ ca un subgrup al lui $GF(q^k)^*$ (Se notează cu K^* grupul multiplicativ $K \setminus \{0\}$).
Acest lucru permite multiplicarea de puncte, lucru imposibil în grupul aditiv $E(Z_q)$.

Mai mult, de obicei gradul de acoperire este un număr foarte mare (aproape acelaşi număr de biţi ca n), deci calculele din $GF(q^k)^*$ sunt de complexitate exponenţială (în q). Din acest motiv, k este cunoscut şi sub numele "*multiplicator de securitate*".

Definiţia 1.2. *([35] pag. 194, [53] pag. 63) Fie E o curbă eliptică peste Z_q, n un număr prim cu q şi P un punct de ordin n al curbei. O "funcţie de distorsiune" (pentru P) este un endomorfism non-raţional care asociază lui P un punct $\phi(P)$ liniar independent de P.*

Exemplul 1.2. *Să considerăm curba $y^2 = x^3 + a$ peste Z_q, unde q este un număr prim, $q \equiv 3 \ (mod\ 4)$ şi $a \in Z_q$. O curbă de distorsiune pe E poate fi de exemplu*

$$\phi(x, y) = (\xi x, y), \qquad unde \qquad \xi = \frac{q-1}{2} \cdot (1 + 3^{(q+1)/4} i).$$

ξ are ordinul 3.
Într-adevăr, $\xi \neq 1$ şi

$$\xi^3 = \left(\frac{q-1}{2}\right)^3 \left(1 - 3^{(q+3)/2} + i \cdot \left(3^{(q+5)/4} - 3^{(3q+3)/4}\right)\right).$$

Folosind teorema lui Euler ($3^{q-1} \equiv 1 \ (mod\ q)$), partea reală a acestui număr este 1, iar partea imaginară este 0.

Dacă luăm acum $q = 11$ şi $a = 1$, curba $\phi(x, y) = (\xi x, y)$, unde

$$\xi = \left(\frac{11-1}{2}\right)\left(1 + 3^{(11+1)/2} i\right) \equiv 5(1 + 5i) \pmod{11}$$

este o curbă de distorsiune pentru punctul $P = (2, 3)$.
Prin calcul, avem $\phi(2, 3) = ((5 + 3i) \cdot 2, 3) = (10 + 6i, 3)$, punct liniar independent de P.

Exemplul 1.3. *Fie curba eliptică $y^2 = x^3 + x$ peste Z_{11} şi $\phi(x, y) = (-x, iy)$. Aceasta este o funcţie de distorsiune pentru $P = (0, 1)$, deoarece $\phi(P) = \phi(0, 1) = (0, i)$, valoare liniar independentă de P.*

1.2 Perechi biliniare

În ultima decadă, în criptografia curbelor eliptice a apărut un aparat de lucru tot mai frecvent folosit: cel de "pereche biliniară".

Pe scurt, o *pereche bilinară*[2] este o funcţie care aplică o pereche de puncte ale unei curbe eliptice într-un element dintr-un grup multiplicativ peste un corp finit. Formal:

Definiţia 1.3. *Fie n un număr întreg pozitiv, G_1, G_2 două grupuri abeliene aditive – pe o curbă eliptică – ambele de caracteristică n (deci $[n]P = 0 \ \forall P \in G_1 \cup G_2$) şi G_3 un grup ciclic multiplicativ de ordin n. O pereche biliniară este o aplicaţie*

[2]Noţiunile definite în această secţiune au ca sursă bibliografică [35] şi [53].

$$e : G_1 \times G_2 \longrightarrow G_3$$

cu proprietăţile:

1. *(Biliniaritate):* $\forall P, P' \in G_1,\ Q, Q' \in G_2$, *avem*

$$e(P + P', Q) = e(P, Q) \cdot e(P', Q), \qquad e(P, Q + Q') = e(P, Q) \cdot e(P, Q')$$

2. *(Non-degenerare):*

   (a) $\forall P \in G_1,\ P \neq 0,\ \exists Q \in G_2$ *cu* $e(P, Q) \neq 1$.

   (b) $\forall Q \in G_2,\ Q \neq 0,\ \exists P \in G_1$ *cu* $e(P, Q) \neq 1$.

Exemplul 1.4. *Definiţia de mai sus nu este specifică curbelor eliptice. Astfel, funcţia* $f : G_1 \times G_1 \longrightarrow G_2$, *unde* $G_1 = (Z_p, +)$, $G_2 = (Z_q, ^*)$ *iar* q *este un număr prim, definită*

$$f(x, y) = 2^x \cdot 3^y$$

este o pereche biliniară, ea verificând imediat condiţiile de biliniaritate şi non-degenerare.

În domeniul securităţii informaţiei însă, această noţiune este utilizată până acum numai în zona curbelor eliptice.

Lema 1.1. *Fie* e *o pereche bilinară şi* $P \in G_1,\ Q \in G_2$. *Atunci*

1. $e(P, 0) = e(0, Q) = 1$.
2. $e(-P, Q) = e(P, Q)^{-1} = e(P, -Q)$.
3. *Pentru orice număr întreg pozitiv* j, $e([j]P, Q) = e(P, Q)^j = e(P, [j]Q)$.

1.2.1 Divizori

Definiţia 1.4. *Fie* f *o funcţie raţională (poate fi reprezentată ca o expresie raţională de polinoame).*

- a *este o rădăcină a lui* f *dacă* $f(a) = 0$.
- a *este un pol al lui* f *dacă* $f(a) = \pm\infty$.
- $f(X)$ *are un pol la infinit dacă* $f(1/X)$ *are un pol în* $x = 0$.
- $f(X)$ *are o rădăcină la infinit dacă* $f(1/X)$ *are rădăcină pe* $x = 0$.

Exemplul 1.5. *Funcţia*

$$f(x) = \frac{(x-1)^2}{(x+2)^3} = (x-1)^2(x+2)^{-3}$$

are pe 1 *ca rădăcină de ordin* 2, *pe* 0 *ca rădăcină de ordin* 1 *la infinit şi pe* -2 *ca pol de ordin* 3.

Definiţia 1.5. *Dacă*

$$f(X) = \prod_i (X - x_i)^{a_i}$$

vom nota cu $div(f)$ *(divizorii lui* f*) suma formală*

$$div(f) = \sum_i a_i(x_i)$$

De remarcat că produsul a două funcţii raţionale are ca efect adunarea sumelor formale, iar câtul a două funcţii – scăderea sumelor formale.

Exemplul 1.6. *Dacă*

$$f(X) = \frac{(X-1)^2}{(X+2)^3}, \qquad g(X) = \frac{(X+2)^3}{(X+1)^4}$$

atunci lui $f(X) \cdot g(X) = (X-1)^2/(X+1)^4$ *îi corespunde*

$div(fg) = div(f) + div(g) = 2(1) + (\infty) - 3(-2) + 3(-2) + (\infty) - 4(-1) = 2(1) + 2(\infty) - 4(-1)$.

Evident, $div(f) = 0$ dacă şi numai dacă f este o constantă. Deci, dacă $div(f) = div(g)$ atunci $div(g/f) = 0$ şi g este un multiplu cu o constantă a lui f. Altfel spus, $div(f)$ determină o funcţie f, abstracţie făcând de înmulţirea cu o constantă nenulă.

Definiţia 1.6. *Fie* E *o curbă eliptică. Un divizor al lui* E *este o sumă formală de forma*

$$D = \sum_{P \in E} n_P(P)$$

unde n_P *sunt numere întregi şi suma are un număr finit de termeni.*
$gr(D) = \sum_{P \in E} n_P$ *se numeşte "gradul" lui* D.

Definiţia 1.7. *Un divizor* D *este "divizor principal" dacă există o funcţie raţională* f *astfel ca* $D = div(f)$.

O formă echivalentă a acestei definiţii este:
Un divizor D este *principal* dacă se poate scrie sub forma $D = \sum_i a_i(P_i)$ unde $gr(D) = 0$ şi $\sum_i a_i P_i = \mathcal{O}$.
În particular, dacă P este un punct de ordin n, atunci $n(P) - n(\mathcal{O})$ este un divizor principal.

Definiţia 1.8. *Fie* E *o curbă eliptică şi un divizor* $D = \sum_{P \in E} n_P(P)$. *Suportul lui* D *este mulţimea punctelor* P *pentru care* $n_P \neq 0$. *Notăm această mulţime cu* $supp(D)$.

Doi divizori D_1 şi D_2 au suporturi disjuncte dacă $supp(D_1) \cap supp(D_2) = \emptyset$.

Exemplul 1.7. *Divizorii* $D_1 = (P) - (\mathcal{O})$ *şi* $D_2 = (P + R) - (R)$ *au suporturi disjuncte cât timp* $\{P, \mathcal{O}\} \cap \{P + R, R\} = \emptyset$.

Divizorii $D_1 = (P) - (\mathcal{O})$ *şi* $D_2 = (Q) - (\mathcal{O})$ *nu au suporturi disjuncte.*

Definiţia 1.9. *Dacă* $D = \sum_i a_i(P_i)$ *este un divizor şi* f *este o funcţie raţională, se defineşte valoarea lui* f *în* D *prin*

$$f(D) = \prod_i f(P_i)^{a_i}$$

Exemplul 1.8.

1. *Dacă* $D = 2(P) - 3(Q)$ *atunci* $f(D) = f(P)^2 \cdot f(Q)^{-3}$.

2. *Dacă* $P = (2, 3)$, $Q = (0, 1)$ *sunt puncte pe o curbă eliptică* E *peste* Z_{11}, $D = (P) - (Q)$ *şi* f *este o funcţie definită* $f(x, y) = y + 1$, *atunci*
$$f(D) = \frac{3+1}{1+1} = 4 \cdot 2^{-1} = 4 \cdot 6 \equiv 2 \pmod{11}.$$

Teorema 1.1. *(Reciprocitate Weil) Fie* f, g *două funcţii raţionale definite peste un corp* F. *Dacă* $div(f)$ *şi* $div(g)$ *au suporturi disjuncte, atunci*

$$f(div(g)) = g(div(f))$$

Exemplul 1.9. *Să considerăm (peste* Z_{11}*) funcţiile raţionale*

$$f(X) = \frac{X-2}{X-7} \quad şi \quad g(X) = \frac{X-6}{X-5}$$

Avem $div(f) = (2) - (7)$, $div(g) = (6) - (5)$. *Atunci*

$$f(div(g)) = \frac{f(6)}{f(5)} = \frac{7}{4} = 7 \cdot 3 \equiv 10 \pmod{11}$$

$$g(div(f)) = \frac{g(2)}{g(7)} = \frac{5}{6} = 5 \cdot 2 \equiv 10 \pmod{11}$$

Definiţia 1.10. *Divizorii* D_1, D_2 *sunt echivalenţi dacă* $D = D_1 - D_2$ *este un divizor principal.*

Exemplul 1.10.

Dacă f *este o funcţie raţională, atunci divizorii* $(P) - (\mathcal{O})$ *şi* $(P) - (\mathcal{O}) + div(f)$ *sunt echivalenţi.*

1.3 Perechi Tate

Utilizarea perechilor Tate în criptografie îşi are originea într-un articol nepublicat din 1986 al lui Victor Miller, continuat cu rezultate ale lui Menezes - Okamoto - Vanstone şi Frey - Rueck.

Fie E o curbă eliptică peste un corp K_0 şi n un întreg pozitiv coprim cu caracteristica lui K_0. Mulţimea rădăcinilor de ordinul n ale unităţii este definită $I_n = \{u \in \overline{K_0}^* \mid u^n = 1\}$ unde $\overline{K}$ reprezintă închiderea algebrică a lui K.

Definim corpul $K = K_0(I_n)$ ca fiind extensia lui K_0 generată de rădăcinile de ordinul n ale unităţii. Fie acum

$$E(K)[n] = \{P \in E(K) \mid [n]P = \mathcal{O}\}$$

şi

$$nE(K) = \{[n]P \mid P \in E(K)\}.$$

Deci

- $E(K)[n]$ este un grup multiplicativ de caracteristică n.
- $nE(K)$ este un subgrup al lui $E(K)$.
- Grupul cât $E(K)/nE(K)$ este un grup de caracteristică n.

Observaţia 1.1. *Se poate considera $E(K)/nE(K)$ ca fiind mulţimea claselor de echivalenţă a punctelor din $E(K)$ faţă de relaţia $P_1 \equiv P_2$ dacă şi numai dacă $P_1 - P_2 \in nE(K)$.*

Dacă notăm $(K^*)^n = \{u^n \mid u \in K^*\}$, evident $(K^*)^n$ este subgrup al lui K^*, iar $K^*/(K^*)^n$ este un grup multiplicativ de caracteristică n.
Mai mult, grupurile $K^*/(K^*)^n$ şi I_n sunt izomorfe.

Fie $P \in E(K)[n]$ şi $Q \in E(K)$ (privit ca reprezentant al unei clase de echivalenţă în $E(K)/nE(K)$). Deoarece $[n]P = \mathcal{O}$, rezultă că există o funcţie raţională f astfel ca $div(f) = n(P) - n(\mathcal{O})$. Fie D un divizor de grad 0, definit peste K şi echivalent cu $(Q) - (\mathcal{O})$ (un astfel de divizor poate fi construit uşor: se ia un punct arbitrar $S \in E(K)$ şi de defineşte $D = (Q + S) - (S)$.).
Mai mult, presupunem că $supp(D)$ şi $supp(div(f))$ sunt disjuncte.

Deoarece f şi D sunt definite peste corpul K, valoarea $f(D) \in K$. Mai mult, deoarece $div(f)$ şi D au suporturi disjuncte, vom avea $f(D) \neq 0$, deci $f(D) \in K^*$.

Perechea Tate a lui P şi A este definită prin

$$e_n(P, Q) = f(D)$$

interpretată ca un element din $K^*/(K^*)^n$.

Exemplul 1.11. *Fie curba eliptică*

$$E: \quad y^2 = x(x^2 + 2x + 2)$$

peste Z_3; deci $E(Z_3) = \{\mathcal{O}, (0,0)\}$. Fie $n = 2$, $P = (0,0)$; se observă că $I_2 = \{1, 2\} \subset Z_3$. Să presupunem că vrem să calculăm $e_2(P, P)$.

Funcţia $f(x) = x$ are $div(x) = 2(P) - 2(\mathcal{O})$. Trebuie să găsim un divizor D echivalent cu $(P) - (\mathcal{O})$ şi cu suport disjunct de $\{\mathcal{O}, P\}$.

Deoarece $E(Z_3)$ are numai două puncte, nu putem construi $D = (P + S) - (S)$ pentru un $S \in E(Z_3)$. Va trebui să procedăm altfel.

Construim $GF(3^2) \approx Z_3(i)$ *unde* $i^2 = -1$ *şi considerăm punctele lui* E *peste* $GF(3^2)$. *Divizorul* $D = ((1+i, 1+i) + ((1-i, 1-i)) - ((1,i)) - ((1,-i))$ *este definit peste* Z_3 *ca divizor. Cum* $(1,i) + (1,-i) = \mathcal{O}$ *în* $E(Z_{3^2})$ *şi* $(1+i, 1+i) + (1-i, 1-i) \neq \mathcal{O}$ *este un punct din* $E(GF(3^2))$, *rezultă că este* P. *Pe baza Definiţiei 1.7 avem că* D *este un divizor principal, echivalent cu* $(P) - (\mathcal{O})$. *Atunci*

$$e_2(P,P) = x(D) = \frac{(1+i)\cdot(1+i)}{1\cdot 1} = 2.$$

Proprietăţi ale perechilor Tate

O pereche Tate nu este un element bine definit din K^*: ea depinde de alegerea lui D şi a reprezentantului Q dintr-o clasă din $E(K)/nE(K)$. Din acest motiv, valorile perechilor Tate sunt considerate clase de echivalenţă din $K^*/(K^*)^n$. Următoarele două leme arată că o pereche Tate este un element bine definit din $K^*/(K^*)^n$.

Lema 1.2. *Fie* f *o funcţie cu* $div(f) = n(P) - n(\mathcal{O})$ *şi* D_1, D_2 *doi divizori echivalenţi ale căror suporturi sunt disjunte de* $supp(div(f))$. *Dacă* f, D_1, D_2 *sunt definite peste* K, *atunci* $f(D_2)/f(D_1) \in (K^*)^n$.

Lema 1.3. *Fie* $P \in E(K)[n]$ *şi* $Q, R \in E(K)$. *Fie* f *o funcţie cu* $div(f) = n(P) - n(\mathcal{O})$ *şi* D_1, D_2 *doi divizori definiţi peste* K *ale căror suporturi sunt disjuncte de* $supp(div(f))$. *Dacă* D_1 *este echivalent cu* $(Q) - (\mathcal{O})$ *şi* D_2 *este echivalent cu* $(Q + [n]R) - (\mathcal{O})$ *atunci* $f(D_2)/f(D_1) \in (K^*)^n$.

Demonstraţiile acestor două leme se găsesc în [35].

- Dacă $div(f) = n(P) - n(\mathcal{O})$ atunci o pereche Tate $e_n(P,Q)$ poate fi definită folosind o funcţie g cu $div(g) = n(P+R) - n(R)$ unde $r \in E(K)$ este arbitrar.
- O pereche Tate este nedegenerată: $\forall P \in E(K)[n] \setminus \{\mathcal{O}\}$ există $Q \in E(K)/nE(K)$ astfel ca $e_n(P,Q) \neq 1$.
- O pereche Tate este biliniară: $\forall P, P_1, P_2 \in E(K)[n]$, $\forall Q, Q_1, Q_2 \in E(K)/nE(K)$,

$$e_n(P_1 + P_2, Q) = e_n(P_1, Q) \cdot e_n(P_2, Q), \qquad e_n(P, Q_1 + Q_2) = e_n(P, Q_1) \cdot e_n(P, Q_2)$$

Un rezultat simplu de demonstrat ([35], pag. 189), dar esenţial pentru criptografie – deoarece arată când valoarea unei perechi Tate poate fi banală – este

Lema 1.4. *Fie* E *o curbă eliptică peste* Z_q *şi* n *un număr prim. Notăm cu* $k > 1$ *gradul de acoperire al curbei* E *în raport cu* n. *Dacă* L *este un corp cu proprietatea* $Z_q \subset L \subset GF(q^k)$ *şi* $P, Q \in E(L)$, *atunci* $e_n(P,Q) \in \left(GF(q^k)^*\right)^n$.

Anexa 2

Problema Diffie - Hellman

De la publicarea sa în 1976, protocolul propus de Whitfield Diffie şi Martin Hellman a devenit unul din cele mai cunoscuite şi utilizate primitive criptografice. În versiunea sa de bază (aşa cum este prezentată în Capitolul 5), el este o soluţie eficientă pentru probleme de partajare a cheilor de sesiune între doi participanţi.

Definiţia 2.1. *Fie G un grup ciclic şi $g \in G$ un generator al său. Fiind date două numere întregi a, b, problema determinării lui g^{ab} din g^a şi g^b se numeşte "problema Diffie Hellman" în raport cu g (DHP).*

O posibilitate de a rezolva DHP este de a calcula întâi a (din g^a) şi b (din g^b), după care se determină imediat g^{ab}. Aceasta reduce DHP la Problema logaritmului discret.

Definiţia 2.2. *Fie G un grup ciclic generat de g. Fiind dat $h \in G$, determinarea unui număr întreg a astfel ca $g^a = h$ se numeşte "problema logaritmului discret" în raport cu g (DLP).*

Rezolvarea celor două probleme este strâns legată de metoda de reprezentare a grupurilor. Astfel, pentru multe grupuri nu se cunoaşte dacă cel mai eficient mod de a rezolva DHP este de a rezolva întâi DLP. De asemenea, nu se ştie dacă există grupuri în care DHP este substanţial mai simplă decât DLP.

Pentru a ne restrânge numai la informaţiile necesare înţelegerii problemei Diffie - Hellman, vom considera grupuri în care DLP este o problemă $\mathcal{NP}$ - completă.

2.1 Variante ale problemei Diffie - Hellman

Definiţia 2.3. *Fie G un grup ciclic generat de g. Fiind date valorile g^a, g^b, problema calculării lui g^{ab} se numeşte "problema Diffie - Hellman computaţională" (CDHP) în raport cu (g, a, b).*

Definiţia 2.4. *Fie G un grup ciclic generat de g şi g^a, g^b, x trei valori din G alese independent şi aleator. Problema de a determina cu o probabilitate semnificativă dacă $x = g^{ab}$ se numeşte "problema Diffie - Hellman decizională" (DDHP).*

Prin *probabilitate semnficativă* se înţelege o probabilitate substanţial mai mare de 1/2.

O modalitate de a rezolva problema este de a calcula g^{ab} rezolvând CDHP, şi apoi de a compara această valoare cu x.

Deci DDHP nu este mai dificilă decât CDHP.

Aparent, DDHD este mai uşoară decât DHP. Astfel, să considerăm grupul multiplicativ Z_p^* cu p prim. Aici calculul simbolului Legendre este uşor. Valoarea g^{ab} este un pătrat perfect modulo p dacă şi numai dacă $a \cdot b$ este par, ceea ce se întâmplă cu probabilitate 3/4 (pentru a şi b generaţi aleator). Pe de altă parte, probabilitatea ca – pentru c generat aleator – $\left(\frac{x}{p}\right) = 1$ (respectiv -1) este 1/2.

Deci probabilitatea ca $\left(\frac{g^{ab}}{p}\right)$ şi $\left(\frac{x}{p}\right)$ să fie diferite este

$$Pr\left(\left(\frac{g^{ab}}{p}\right) = 1 \wedge \left(\frac{x}{p}\right) = -1\right) + Pr\left(\left(\frac{g^{ab}}{p}\right) = -1 \wedge \left(\frac{x}{p}\right) = 1\right) = \frac{3}{4} \cdot \frac{1}{2} + \frac{1}{4} \cdot \frac{1}{2} = \frac{1}{2}.$$

Deci, comparând simbolurile Legendre $\left(\frac{g^{ab}}{p}\right)$ şi $\left(\frac{x}{p}\right)$ sunt doar 25% şanse de a distinge între g^{ab} şi o valoare aleatoare c.

Deşi DDHP este uşoară în Z_p^* cu p prim, se consideră că ea este dificilă în $GF(p^k)^*$ cu $k > 1$.

O relaţie între aceste probleme este dată de schema următoare:

S-a notat $A \longrightarrow B$ faptul că o soluţie la A determină o soluţie la B.

2.2 Problema Diffie - Hellman pe curbe eliptice

Fie $E(K)$ grupul aditiv al punctelor unei curbe eliptice (Anexa 1). Atunci CDHP se poate formula astfel (pe $E(K)$):

Fiind date $P, aP, bP \in E(K)$**, să se calculeze** abP**.**

Definiţia următoare generalizează CDHP la grupuri cu perechi biliniare:

Definiţia 2.5. *Fie grupul aditiv $G_1 \subseteq E(K)$, grupul multiplicativ G_2 sî o pereche biliniară $e : G_1 \times G_1 \longrightarrow G_2$. Fiind date $P, aP, bP, cP \in G_1$, determinarea valorii $e(P, P)^{abc}$ se numeşte "Problema Diffie - Hellman biliniară" (BDHP).*

Evident, BDHP poate fi scrisă şi în notaţie multiplicativă: fiind date g, g^a, g^b, g^c, se cere calculul lui $e(g,g)^{abc}$.

A rezolva BDPH nu este mai simplu decât DLP în G_1 sau G_2:

- Dacă putem să aflăm valoarea lui c calculând logaritmul discret al lui cP în G_1, atunci vom determina $e(aP,bP)^r = \left(e(P,P)^{ab}\right)^c = e(P,P)^{abc}$.
- Dacă putem determina valoarea lui c calculând logaritmul discret al lui $e(P,cP) = e(P,P)^c$ în G_2, atunci vom calcula $e(P,P)^{abc}$ în mod similar.

De remarcat că aplicaţia $\phi_P : G_1 \longrightarrow G_2$ definită $\phi_P(Q) = e(P,Q)$ este un izomorfism de grupuri.

Corolarul 2.1. *Dacă inversa lui ϕ_P se calculează uşor, atunci*

1. *BDHP este uşoară în G_1.*
2. *DDHP este uşoară în G_2*

Demonstraţie.

1. Se calculează întâi $\alpha = e(aP,bP) = e(P,abP)$, apoi $\phi_P^{-1}(\alpha) = abP$. În final se determină $e(abP,cP) = e(P,P)^{abc}$.
2. Fie $g, g^a, g^b, x \in G_2$, unde g este un generator. Dacă $\phi_P^{-1}(g) = Q$, vom avea $\phi_P^{-1}(g^a) = aQ$ şi $\phi_P^{-1}(g^b) = bQ$.
   Să presupunem că $\phi_P^{-1}(x) = X$. Dacă $x = g^{ab}$, atunci $\phi_P^{-1}(x) = abQ$, deci $e(Q,X) = e(Q,abQ) = e(Q,Q)^{ab} = e(aQ,bQ)$, adică $x = g^{ab}$.

□

Chiar dacă este dificil pentru *Oscar* să determine $e(P,P)^{abc}$ din P, aP, bP şi cP, nu există nici o garanţie că el nu poate deduce anumite informaţii parţiale despre $e(P,P)^{abc}$ (de exemplu, să afle o parte din biţi). Pentru a elimina o astfel de posibilitate, o altă problemă trebuie să fie grea: problema de decizie Diffie - Hellman biliniară:

Definiţia 2.6. *Fie E o curbă eliptică peste un corp K, $e : G_1 \times G_1 \longrightarrow G_2$ o pereche biliniară (unde $G_1 \subseteq E(K)$) şi $P, aP, bP, cP, x \in G_1$. Problema de a determina cu probabilitate semnificativă dacă $x = e(P,P)^{abc}$ se numeşte "Problema de decizie Diffie - Hellman bilinară" (DBDHP).*

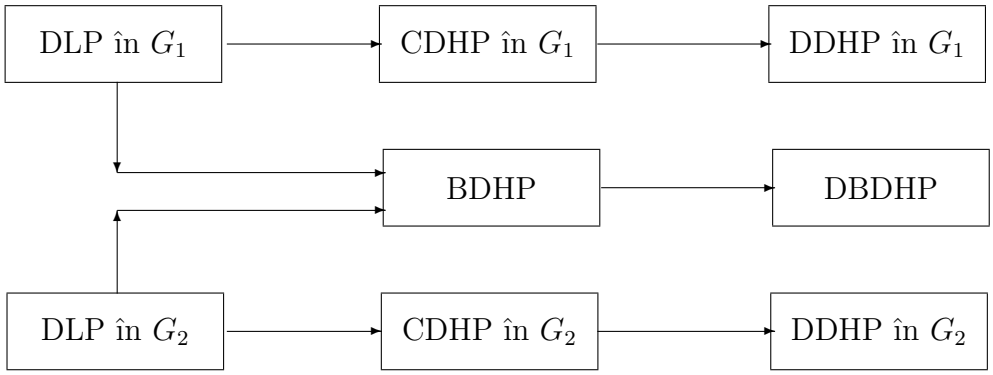
Rezolvarea DBDHP nu este mai dificilă decât DLP în G_1 sau G_2. Într-adevăr:

1. Dacă se poate afla c calculând logaritmul discret al lui cP în G_1, atunci se poate determina
$$e(aP,bP)^c = \left[e(P,P)^{ab}\right]^c = e(P,P)^{abc}$$

2. Dacă se poate afla c calculând logaritmul discret al lui $e(P, cP) = e(P, P)^c$ în G_2, atunci se poate calcula în mod similar şi $e(P, P)^{abc}$.

Dacă DBDHP este grea, atunci este greu de distins între $e(P, P)^{abc}$ şi orice alt element din G_2; deci $e(P, P)^{abc}$ va arăta ca un element oarecare din G_2.

Figura următoare arată relaţia dintre diverse variante ale problemei Diffie - Hellman şi DLP.



2.3 Probleme Diffie - Hellman co-biliniare

Dacă se foloseşte o pereche biliniară $e : G_1 \times G_2 \longrightarrow G_3$ cu $G_2 \neq G_2$, toate problemele Diffie - Hellman biliniare trebuie reformulate.

Definiţia 2.7. *Fie E o curbă eliptică peste un corp K, G_1, G_2 subgrupuri distincte din $E(K)$ şi $e : G_1 \times G_2 \longrightarrow G_3$ o pereche biliniară. Fiind date $P, aP, bP \in G_1$, $Q \in G_2$, problema calculări valorii $e(P, Q)^{ab}$ se numeşte "Problema Diffie - Hellman co-biliniară" (co-BDHP).*

În mod similar se pot reformula şi celelalte probleme în variantă co-biliniară. Toate implicaţiile arătate în această anexă rămân valabile şi în cazul co-biliniarităţii.

Multe lucrări legate de problemele Diffie - Hellman şi perechi biliniare se referă de fapt la noţiunea de co-biliniaritate.

Bibliografie

[1] C. A. Asmuth, J. Bloom – *A modular approach to key safeguarding*, IEEE Trans on IT 29 (1983), pp. 208-210.

[2] Atanasiu, A. – *Securitatea Informaţiei, vol. 1, Criptografie*, ed. InfoData, Cluj, 2008.

[3] D. Bayer, S.Haber, W.Stornetta – *Improving the efficiency and reliability of digital time-stamping. Sequences II*, Methods in Communication, Security and Computer Science, Springer Verlag (1993), 329-334.

[4] J. Benaloh – *Verifiable sectret-ballot elections*, Ph.D thesis, Yale University, Technical report 561, 1987.

[5] J. Benaloh, J. Leichter – *Generalized secret sharing and monotone functions*, în "Advances in Cryptology – CRYPTO 8", S. Goldwasser, ed., LNCS 403 (1989), pp. 27-35.

[6] S. Blake - Wilson, A. Menezes – *Authenticated Diffie - Hellman Key Agreement Protocols*, Proc. of the 5-th Intern. Workshop on Security Protocols, LNCS 1361, 1997, pp. 137-158.

[7] G. R. Blakley – *Safeguarding cryptographic keys*, în "Proc. of the National Computer Conference, 1979", American Federation of Information Processing Societies Proc. 48 (1979), pp. 313-317.

[8] Boneh, D., Franklin, M. – *Identity Based Encryption from the Weil Pairing*, SIAM Journal of Computing, vol. 32, no. 3, pp. 586-615.

[9] Boneh, D., Boyen, X. – *Efficient Selective - ID Secure Identity Based Encryption Without Random Oracles*, Proc. of EUROCRYPT 2004, Interlaken, Elveţia, 2-6 May 2004, pp. 223-238.

[10] J. N. Bos, D. Chaum - *Provably unforgable signatures*, LNCS, 740 (1993), pp. 1-14.

[11] S. Brands – *An Efficient Off-Line Electronic Cash System Based On The Representation Problem*, Technical Report CS-R9323, 1993, CWI, Amsterdam, Netherlands.

[12] D. Chaum – *Untraceable electronic mail, return addresses, and digital pseudonyms*, Commun. ACM, 24 (2) (1981), pp. 84 - 90.

[13] D. Chaum, H. van Antwerpen – *Undeniable signatures*, LNCS, 435 (1990), pp. 212-216.

[14] D. Chaum, A. Fiat, M. Naor – *Untraceable Electronic Cash*, Advances in Cryptology, CRYPTO 8, S. Goldwasser (Ed.), Springer-Verlag, pp. 319-327.

[15] D. Chaum, E. van Heyst – *Group signatures*, Advances in Cryptology, EUROCRYPT 91, LNCS, vol. 547, Springer-Verlag (1991), pp. 257-265.

[16] D. Chaum, E. van Heijst, B. Pfitzmann – *Cryptographically strong undeniable signatures, unconditionally secure for the signer*, LNCS, 576 (1992), pp. 470-484.

[17] Chen, L. ş.a – *An Efficient ID-KEM Based on the Sakai-Kasahara Key Construction*, IEEE Proc. Information Theory, vol. 153, no. 1, (2006), pp. 19-26.

[18] T-S Chen, K-H Huang, Y-F Chung – *Digital Multi - Signature Scheme based on the Elliptic Curve Cryptosystem*, J. Comp. Sci & Technol. vol. 19 (2004), no. 4, pp. 570-573.

[19] X. Chen, F. Zang, K. Kim – *A new ID - based group signature scheme from bilinear pairings*, *http* : *//eprint.iacr.org*/2003/100.*pdf*, 2003.

[20] B. Chor, S. Goldwasser, S. Micali, B. Awerbuch – *Verifiable secret sharing and achieving simultaneity in the presence of faults*, Proc. of the 26th IEEE Symposium on the Foundations of Computer Science, (1985), pp. 383-395

[21] Cocks, C. – *an Identity Based Encryption Scheme Based on Quadratic Residues*, Proc. of the Eighth IMA Intern. Conf. on Cryptography and Coding, Cirencester, 17-19 Dec. 2001, pp. 360-363.

[22] J.S. Coron, D. Naccache, J. Stern – *On the security of RSA Padding*, In Advances of Cryptology CRYPTO 99, LNCS 1666, Springer - Verlag, 1999, pp. 1-18.

[23] R. Cramer, R. Gennaro, B. Schoenmakers – *A secure and optimally efficient multiauthority election scheme*, în EUROCRYPT 1997, pp. 103 118.

[24] I.B. Damgard –*A design principle for hash functions*, LNCS, 435 (1990), pp. 516-427.

[25] H. Delfs, H. Knebl – *Introduction to Cryptography, Second edition*, Springer Verlag, 2007.

[26] W. Diffie, M.E. Hellman – *Multiuser cryptographic techniques*, AFIPS Conference Proceedings, 45 (1976), pp. 109 - 112.

[27] W. Diffie, M.E. Hellman – *New Directions in Cryptography*, IEEE Trans. on Information Theory, vol. IT-22 (1976), pp. 644 - 654

[28] H. Dobbertin – *Cryptanalysis of MD4*, Journal of Cryptology, 11 (1998), pp. 253-271.

[29] T. ElGamal – *A public key cryptosystem and a signature scheme based on discrete algorithms*, IEEE Trans. on Information Theory, 31 (1985), pp. 469-472.

[30] M. J. Farsi – *Digital Cash*, Masters Thesis in Computer Science, Dpt of Mathematics and Computing Science, Goteborg University 1997.

[31] P. Feldman – *A practical scheme for non-interactive verifiable secret sharing*, Proc. of the 28th IEEE Symposium on the Foundations of Computer Science, (1987), pp. 427-437.

[32] N. Ferguson – *Single Term Off-Line Coins*, Advances in Cryptology - EUROCRYPT 93, Springer-Verlag, pp. 318-328.

[33] A. Fujioka, T. Okamoto, K. Ohta – *A practical secret voting scheme for large scale elections*, în J. Seberry şi Y. Zheng, (eds), ASIACRYPT, LNCS 718 (1992), pp. 244 251.

[34] E. Fujisaki, T. Okamoto – *Secure Integration of Assymetric and Simmetric Encryption Schemes*, Proc. of Crypto 99, Santa Barbara, CA, August 20-24, 1999, pp. 537-554.

[35] S. Galbraith – *Pairings*, în "Advances in Elliptic Curve Cryptography" ed. T. Blake, G. Seroussi şi N. Smart; London Math. Society, Lecture Notes Series 317 (2005), pp. 183-214.

[36] T. El Gamal – *A public key cryptosystem and a signature scheme based on discrete algorithms*, IEEE Trans on Inf. Theory, 31 (1985), pp. 469-472.

[37] J. Gibson – *Discrete logarithm hash function that is collision free and one way*, IEEE Proceedings-E, 138 (1991), pp. 407-410.

[38] H. Ghodosi, J. Pieprzyk, Safavi-Naini – *Remarks on the multiple assignment secret sharing scheme*, LNCS 1334 (1997), pp. 72-82.

[39] S. Haber, W. Stornetta; How to timestamp a digital document. Journal of Cryptology, 3 (1991), pp. 99-111.

[40] E. van Heyst, T.P.Petersen – *How to make efficient fail-stop signatures*, LNCS, 658 (1993), pp. 366-377.

[41] S. Iftene – *A generalisation of Mignottes secret sharing scheme*, în Proc. of the 6th Intern. Symposium on Symbolic and Numeric Algorithms for scientific computing, Timişoara, T. Jebelean, V. Negru, D. Petcu, D. Zaharia (eds), Sept. 2004, pp. 196-201, Mirton Publ. House (2004).

[42] I. Ingermarsson, G. D. Simmons – *A protocol to set up shared secret schemes without assistance of mutually trusted party*, EUROCRYPT 90, LNCS vol. 473, Springer - verlag 1991, pp. 266-282

[43] W. Jackson, K.M. Martin, C. M. O′ Keefe – *Muttualy trusted authority - free secret sharing schemes*, Journal of Cryptology, 10 (4), 1997, pp. 261-289.

[44] E. D. Karnin, J. W. Greene, M. E. Hellman – *On secret sharing systems*, IEEE Trans. on Information Theory 29 (1983), pp. 35-41.

[45] V. Klima – *Tunnels in Hash Functions: MD5 Collisions within a Minute*, Cryptology ePrint Archive, http://eprint.iacr.org, Report 105 (2006).

[46] Klitz, E. – *On the Limitations of the Spread of an IBE-to-PKE Transformation*, Proc of PKC 2006, LNCS 3958, Springer, 2006, pp. 274-289.

[47] H. Krawczyk, M. Bellare, R. Canetti – *HMAC: Keyed - Hashing for Message Authentication*, RFC 2104, 1997.

[48] E. Kranakis – *Primality and Cryptography*, Wiley-Teubner Series in Computer Science, 1986.

[49] H. Krawczyk – *Secret sharing made short*, în Advances in Cryptology – CRYPTO 93, D. R. Stinson, ed., LNCS 773 (1994), pp. 136-146.

[50] L. Law, S. Sabett, J. Solinas – *How to make a mint: The Cryptography of Anonymous Electronic Cash*, http://jya.com/nsamint.htm

[51] C.I. Lei, C.I. Fan – *A universal single-authority election system*, IEICE Transactions on Fundamentals E81-A (10) (1998), pp. 2186-2193

[52] Mambo, Masahiro, Keisyke, Usuda, Okamoto – *Proxy signatures: Delegation of the power to sign mesages*, IEICE Trans. Fundamentals, ET9-A (1996), pp. 1338 - 1354.

[53] Martin, L – *Introduction to Identity – Based Encryption*, Artech House, Information Security and Privacy Series, 2008.

[54] T. Matsumoto, Y. Takashima, H. Imai – *On seeking smart public-key distribution systems*, The Trans. of the IECE of Japan, E69 (1986), pp. 99-106.

[55] C. Meadows – *Some threshold schemes without central key distributors*, Congressus Numerantium, 46 (1985), pp. 187-199.

[56] R. J. McEliece, D. Sarwate – *On sharing secrets and Reed-Solomon codes*, Comm. of the ACM 24 (1981), pp. 583-584.

[57] A. Menezes, P. van Oorschot, S. Vanstome – *Handbook of Applied Cryptography*, CRC Press Inc, 1997.

[58] R.C. Merkle – *A fast software one-way functions and DES*, LNCS, 435 (1990), pp. 428-446.

[59] M. Mignotte – *How to share a secret*, în Cryptography Proc., Burg Feuerstein 1982, T. Beth, ed., LNCS 149 (1983), pp. 371-375.

[60] C. J. Mitchell, F. Piper, P. Wild – *Digital signatures*, Contemporary Cryptology, The Science of Information Integrity, IEEE Press, (1992), pp. 325-378.

[61] Y.Mu, V. Varadharajan – *Anonymous e-voting over a network*, Proc. of the 14th Annual Computer Security Applications Conference, ASAC8 (1998) pp. 293-299

[62] R. Needham, M. Schroeder – *Using encryption for authentication in large networks of computers.*, Comm. of the ACM 21, 12 (1978), pp. 993 999.

[63] A. M. Odlyzco – *Cryptanalytic attacks on the multiplicative knapsack cryptosystems and on Shamirs fast signature scheme*, IEEE Trans. on Information Theory, IT30 (1984), pp. 594-601.

[64] T. Okamoto – *Receipt - free electronic voting scheme for large scale elction*, Proc. of Workshop on Security Protocols, LNCS 1361, 1997.

[65] T. Okamoto, K. Ohta – *Universal electronic cash*, J. Feigenbaum (Ed.), Advances in cryptology CRYPTO91, LNCS 576, Springer-Verlag, 1992.

[66] B. Preneel, R. Govaerts, J. Vandewalle – *Hash functions based on block ciphers: a syntetic approach*, LNCS, 773 (1994), pp. 368-378.

[67] M.O. Rabin – *Efficient dispersal of information for security, load balancing, and fault tolerance*, Journal of ACM, 36(2), 1989, pp. 335-348.

[68] R.L. Rivest – *The MD4 message digest algorithm*, LNCS, 537, (1991), pp. 303-311.

[69] R. L. Rivest – *The MD5 Message Digest Algorithm*, RFC 1321, 1992.

[70] A. Salomaa – *Criptografie cu chei publice*, Ed. Militară, 1994.

[71] B. Schoenmakers – *A simple publicly verifiable secret sharing scheme and its application to electronic voting*, Advanced in Cryptology, LNCS 1666 (1999), pp. 148-164.

[72] A. Shamir – *Identity-Based Cryptosystems and Signature Schemes*, Proc. of CRYPTO 84, Santa Barbara, CA. August 19-22, 1984, pp. 47-53.

[73] A. Shamir – *How to share a secret*, Comm of the ACM 22 (1979), pp. 612-613.

[74] G.J. Simmons – *The prisoners problem and the subliminal channel*, Advances in Cryptology - Crypto, 83 (1984), pp. 51-67.

[75] M. E. Smid, D. K. Branstad – *Response to comments on the $NIST$ proposed digital signature standard*, LNCS, 740 (1993), pp. 76-88.

[76] M. Stadler – *Publicly verifiable secret sharing*, Advances in Cryptology - EUROCRYPT 96, LNCS vol. 1070, Springer verlag (1996), pp. 190-199.

[77] D. R. Stinson – *Decomposition constructions for secret sharing schemes*, IEEE Trans. on Information Theory 40 (1994), pp. 118-125.

[78] D. Stinton – *Cryptographie, theorie et pratique*, Intern. Thompson Publ. France, 1995.

[79] Z. Tan, Z. Liu, C. Tan – *Digital proxy Blind Signature Schemes based on DLP and ECDLP*, MM Research Preprints, 212-217, Academia Sinica, Beijing, no. 21, Dec. 2002.

[80] S. Vaudenay – *A Classical Introduction to Cryptography*, Springer Verlag 2006.

[81] H.C.Williams – *Some public-key criptofunctions as intractable as factorisation*, Cryptologia, 9 (1985), pp. 224-237.

[82] ISO/IEC 9796; Information technology – Security Techniques – Digital Signature Scheme Giving message recovery, Intern. Organisation for Standardisation, Geneva, 1991.

[83] *Secure Hash Standard*, National Bureau of Standards, FIPS Publications 180, 1993.

[84] *SKIPJACK and KEA Algorithm Specifications*, versiunea 2.0, http://csrc.nist.gov/groups/ST/toolkit/documents/skipjack/skipjack.pdf

[85] *Digital signature standard*, National Bureau of Standards, FIPS Publications 186, 1994

Index