

# TODO

## Undergraduate Research Opportunities

Dragoș Alin Rotaru

Universitatea din Bucuresti, Romania  
r.dragos0@gmail.com

**Abstract.** None

**Keywords:** securitate, scheme de partajare

## 1 Introducere

### 1.1 Istoric

Termenul de criptografie este definit in dictionarul Oxford ca fiind "arta de a scrie si a rezolva coduri". Criptografia moderna s-a desprins de cea clasica in jurul anilor '80, motivand implementarea rigurozitatii matematice pentru definirea constructiilor criptografice. Asta pentru ca in anii anteriori, experienta a dovedit nesiguranta metodelor de criptare, criptanaliza lor fiind uneori triviala (cifrul lui Cezar, Vigenere ??, ??) sau uneori atinsa cu ceva mai mult efort precum Enigma si alte metode din cel de-al doilea razboi mondial. ??

Criptografia moderna se gaseste pretutindeni in viata de zi cu zi de la ATM-uri, cartele telefonice la semnaturi digitale, protocoale de autentificare, licitatii electronice sau bani digitali, luand amploare o data cu aparitia sistemelor cu cheie publica. O definitie potrivita ar fi "studiul stiintific al tehnicilor pentru a securiza informatia digitala, tranzactiile si calculul distribuit.". [1]

### 1.2 Motivatie

TODO nu are legătură: faptul ca sunt scheme cuantice nu are impact asupra structurii de acces; e ok sa menționezi, dar atunci faci asta în secțiunea anterioară, când poți să adaugi și alte modalități de definire: scheme bazate pe latici, scheme bazate pe perechi biliniare, scheme cuantice, etc. Dezavantajul schemelor generale de partajare este dimensiunea componentelor, exponentiala in functie de numarul de participanti. [8] De asemenea, s-au dezvoltat scheme pentru modele de calcul neconventional, cum ar fi cel cuantic. [9]

### 1.3 Structura

TODO

### 1.4 Securitatea Teoretică a Informației

În cazul unor criptosisteme acestea nu pot fi compromise chiar dacă adversarul dispune de o putere computațională nelimitată. Câteva exemple de criptosisteme care garantează securitatea teoretică-informațională sunt: schemele de partajare, unele protocoale multi-party computation, preluarea într-un mod sigur (securizat?) informații de la baze de date. Securitatea teoretică vine însă cu un cost: efortul computațional depus este mult mai mare decât în cazul schemelor care nu garantează securitatea teoretică (se bazează pe dificultatea computațională unor probleme cunoscute). [2]

## 2 Scheme de partajare

În acest capitol se prezintă modelul tradițional de partajare a secretului atunci când o schemă nu este perfectă.

**TODO Diacritice!**

O schemă de partajare constă în distribuirea unui obiect - digital nu se distribuie obiecte, o informație secretă  $\mathcal{S}$  la mai mulți participanți astfel încât orice superset de participanți predefinit înainte (o structură de acces pe care o vom denumi  $\mathcal{A}$ ) să poată reconstitui secretul  $\mathcal{S}$ . Formal, o schemă de partajare este reprezentată de o pereche de algoritmi  $(Gen, Rec)$ :

- $Gen(\mathcal{S}, m)$  este un algoritm care primește la intrare un secret  $\mathcal{S}$  și întoarce un set de componente  $s_1, s_2, \dots, s_m$ .
- $Rec(s_{i_1}, s_{i_2}, \dots, s_{i_q})$  este un algoritm care primește ca parametri de intrare o mulțime de componente și întoarce  $\mathcal{S}$  dacă mulțimea  $s_{i_1}, s_{i_2}, \dots, s_{i_q} \in \mathcal{A}$ .

Majoritatea schemelor constau în mai multe etape precum:

- *Initializare*. Presupune initializarea variabilelor de mediu necesare.
- *Generare*. O entitate autorizată (numită dealer)  $\mathcal{D}$  folosește algoritmul  $Gen$  pentru a genera componentele.
- *Distributie*. Componentele sunt trimise participanților cu ajutorul unui mijloc de comunicare sigur, fără ca acestea să fie vizibile unui atacator.
- *Reconstrucție*. Dându-se o mulțime de componente, se folosește algoritmul  $Rec$  pentru a recupera secretul  $\mathcal{S}$ .

Acestea se clasifică în funcție de cantitatea de informație secretă pe care o pot obține persoanele care nu fac parte din  $\mathcal{A}$ :

- *Sisteme perfecte de partajare*: componentele nu oferă nici o informație teoretică despre  $\mathcal{S}$  indiferent de resursele computaționale.
- *Sisteme statistic sigure*: o fracțiune de informație este dezvăluită despre  $\mathcal{S}$  independent de puterea computațională a adversarului.
- *Sisteme computațional-sigure de partajare*: se bazează pe faptul că reconstituirea lui  $\mathcal{S}$  se reduce la o problemă *dificilă* [?] în lipsa unor informații oferite doar grupului de acces  $\mathcal{A}$ .

În continuare vom prezenta câteva sisteme perfecte de partajare pentru a forma fundatiile utilizate de arhitecturi pentru stocarea fișierelor, disponibile pe o durată îndelungată.

## 2.1 Istoric

Primele scheme de partajare au fost dezvoltate independent de Shamir si Blakley in 1979 [4, 5].

Denumite si scheme majoritare  $(k, n)$ , acestea rezolvau cazul in care oricare grup de participanti cu un cardinal mai mare sau egal decat  $k$  (marimea pragului) poate reconstitui secretul  $\mathcal{S}$  din componentele primite de la dealer. Daca schema este perfect *sigura* atunci oricare grup cu un numar de participanti mai mic decat  $k$  nu obtine vreo informatie despre  $\mathcal{S}$ .

Notam  $P = \{P_1, \dots, P_n\}$  multimea formata din cei  $n$  participanti intr-o schema si  $y \leftarrow^R Y$  ca  $y$  este un element ales uniform aleator din multimea  $Y$ .

Alte scheme de partajare bazându-se pe structuri de acces generale au fost dezvoltate de Ito, Saito, si Nishizeki, realizand o generalizare a schemei Shamir. [6] Benaloh si Leichter au demonstrat ca schemele de partajare de tip prag nu pot fi folosite pe structuri general monotone (familie de submultimi ale lui  $P$ ) si obtin o constructie mai eficienta ca Ito et. al [6] din punct de vedere al numarului de componentel distribuite participantilor. [7]

Autorii prezinta un exemplu trivial pentru care schema Shamir este insuficienta. Consideram cazul in care vrem sa partajam un secret unor între 4 participanti:  $P_1, P_2, P_3, P_4$  astfel incat  $\{P_1, P_2\}$  și  $\{P_3, P_4\}$  să fie singurele mulțimi autorizate pentru reconstrucția secretului  $\mathcal{S}$  (i.e.  $\mathcal{A} = \{\{P_1, P_2\}, \{P_3, P_4\}\}$ ).

## 2.2 Schema unanima

Presupunând ca vrem sa împărțim un secret  $\mathcal{S}$  la  $n$  participanți astfel încât  $\mathcal{S}$  sa poată fi recuperat doar daca cei  $n$  participanți combina componentele pe care le dețin. Metoda este echivalentă cu o schemă  $(n, n)$  majoritară. Un exemplu este schema introdusa de Karin, Greene si Hellman [10] (Fig.)

## 2.3 Schema Shamir

Schema Shamir ofera mai multa flexibilitate ca schema unanima prin faptul ca oricare  $k$  (sau mai multi) participanti din cei  $n$  pot recupera  $\mathcal{S}$ , insa mai puțin de  $k$  nu obtin nicio informatie despre  $\mathcal{S}$ . Schema Shamir este deci o schemă  $(k, n)$  majoritară.

Intuitiv, avand  $k$  puncte in plan  $(x_i, y_i)$ ,  $x_i \neq x_j$   $i, j \in \{1, 2, \dots, k\}$   $\forall i \neq j$ , exista o curba polinomiala unica care trece prin ele. In schimb, pentru a defini o curba polinomiala de grad  $k$  care contine  $k - 1$  puncte date, exista o infinitate de solutii. Evident, orice submultime de valori  $s_i$  de marime egala cu  $k$  este suficienta si necesara pentru a reconstrui polinomul  $f$ . Dupa interpolarea share-urilor detinute de cel puțin  $k$  dintre participanti, secretul  $\mathcal{S}$  se va afla in  $f(0)$ . (Fig. 2)

Pentru un atacator care detine chiar si  $k - 1$  valori  $\mathcal{S}_i$ , acesta nu determină nimic despre  $\mathcal{S}$ , spațiul de soluții posibile fiind identic fata de situatia in care nu reuseste sa obțină vreo componentă.

**Initializare:**

- Fie  $S \in Z_p$  unde  $q > 1$  si  $q$  prim;
- Fie  $n$  numarul de participanti;

**Generare:** Dealerul  $\mathcal{D}$ :

- Alege  $n - 1$  valori aleatoare  $s_i \leftarrow^R Z_p, i \in \{1, 2, \dots, n - 1\}$ ;
- $s_n = S + \sum_{i=1}^{n-1} s_i \pmod{q}$ ;

**Distributie:** Dealerul  $\mathcal{D}$ 

- transmite în mod sigur participantului  $P_i$  componenta  $s_i, i \in \{1, 2, \dots, n\}$ ;

**Reconstructie:** Cei  $n$  participanti:

- Calculeaza  $S = \sum_{i=1}^n s_i \pmod{q}$ .

**Fig. 1.** Schema unanima [10]

## 2.4 Schema Ito, Saito, si Nishizeki

În continuare vom descrie modalitatea de distribuire a componentelor de la care au pornit Ito, Saito si Nishizeki pentru ca schema sa aiba o structura de acces  $\mathcal{A} \subseteq 2^P$  (submulțime a supersetului de participanți) a.i.  $\forall A \in \mathcal{A}, A \subseteq A' \Rightarrow A' \in \mathcal{A}$ . Având construcția unei scheme majoritare  $(k, n)$  autorii au reușit să descrie elementele din  $\mathcal{A}$  folosind rezultatul unei reuniuni de mulțimi de componente cu un număr de elemente mai mare sau egal decât  $k$ . Definim  $x : Pr, x$  are proprietatea  $Pr$ . Fig. 3 detaliază construcția.

Dezavantajul acestei structuri este numărul de componente necesar pentru o structura de acces oarecare  $\mathcal{A}$ . Un mod simplu de construire al funcției *Assign* este Pentru mai multe informații despre funcția *Assign*, cititorul interesat poate citi în [6]. **TODO prezentarea trebuie să fie de sine stătătoare, trebuie măcar să explici în cuvinte ce înseamnă**

## 3 Sisteme de stocare de lunga durata

În această secțiune vom arăta câteva întrebări ale schemelor de partajare. Considerăm cazul în care vrem să stocăm rapoarte medicale, imagini, documente clasificate pe un timp îndelungat într-un mediu electronic. Pe parcursul timpului, pot apărea în schimb, diverse probleme precum dezastre naturale, defecțiunea unor componente hardware, eroare umană, etc. [11] Un sistem de stocare necesar nevoilor noastre trebuie să satisfacă cel puțin următoarele 3 condiții:

**Initializare:**

- Fie  $S \in Z_q$  unde  $q > 1$  si  $q$  prim;
- Fie  $n$  numărul de participanți a.i  $q > n$ ;
- Fie  $k$  numărul minim de componente puse in comun pentru a determina pe  $S$ ;

**Generare:** Dealerul  $\mathcal{D}$ :

- Alege  $n$  valori distincte  $x_i \leftarrow^R Z_q, i = 1, 2, \dots, n$ ;
- Alege  $a_i \leftarrow^R Z_q, i \in \{1, 2, \dots, k-1\}, a_{k-1} \neq 0$ ;
- Construiește polinomul  $f(x) = a_{k-1}x^{k-1} + a_{k-2}x^{k-2} + \dots + a_1x + S$ ;
- Calculează  $s_i = f(x_i) \ i \in \{1, 2, \dots, n\}$ ;

**Distributie:** Dealerul  $\mathcal{D}$ :

- Transmite participantului  $P_i$  componenta  $s_i, i \in \{1, \dots, n-1\}$ ;

**Reconstrucție:** Orice mulțime cu dimensiunea  $k$  (sau mai mare) de participanți distincți  $P_1, P_2, \dots, P_k$ :

- Interpolează punctele  $s_i$  pentru a obține polinomul  $f$ :

$$f(x) = \sum_{i=1}^k s_i \prod_{1 \leq j \leq k, j \neq i} \frac{x - x_j}{x_i - x_j} \quad (1)$$

- Află secretul reconstruit  $S = f(0)$ .

**Fig. 2.** Schema Shamir [5]

- Disponibilitatea: Informatia trebuie sa ramana accesibila tot timpul, in ciuda erorilor de tip hardware.
- Integritatea: Abilitatea sistemului de a raspunde cererilor intr-un mod care garanteaza corectitudinea lor.
- Confidentialitatea: O persoana care nu face parte din grupul de acces sa nu obtina permisiunea de a afla informatii de orice fel despre datele existente in sistem.

### 3.1 Criptare VS scheme de partajare

Una dintre solutiile existente pentru a construi acest sistem ar putea fi criptarea datelor folosind o cheie inainte de inserarea lor in spatiul de stocare. In momentul in care un user autorizat doreste sa efectueze o citire a unor date, intrebuinteaza cheia potrivita pentru a le decripta. In practica exista algoritmi de criptare eficienti precum AES însă aceștia nu garantează confidențialitatea datelor în cazul în care avem de a face cu un adversar fara o limita computationala. Un dezavantaj al criptarii este adminstrarea cheilor, standardele de securitatea schimbându-

**Initializare:**

- Fie  $q$  un numar prim  $q, q > 1, z \in \mathbb{N}$  nenul și  $\mathcal{C} = GF(p^z)$ ;
- Fie  $S \in \mathcal{C}$  secretul;
- Fie structura de acces  $\mathcal{A}$ ;
- Fie  $n$  numarul de participanti;

**Generare:** Dealerul  $\mathcal{D}$ :

- Alege  $n$  valori distincte  $x_i \leftarrow^R Z_q, i = 1, 2, \dots, n$ ;
- Alege  $a_i \leftarrow^R \mathcal{C} \setminus \{0\}, i \in \{1, 2, \dots, k-1\}, a_{k-1} \neq 0$ ;
- Construiește polinomul  $f(x) = a_{k-1}x^{k-1} + a_{k-2}x^{k-2} + \dots + a_1x + S$ ;
- Atribue  $s_i = f(x_i) \ i \in \{1, 2, \dots, n\}$ ; Fie  $Shares = \{s_1, \dots, s_n\}$ ;
- Alege  $D_i \subseteq Shares \ 1 \leq i \leq n$ ;
- Alege functia  $Assign : P \rightarrow 2^Q$ :
  - $Assign(P_i) = D_i \ 1 \leq i \leq n$
  - $\mathcal{A} = \left\{ Q \subseteq Shares : \left| \bigcup_{P_i \in Q} Assign(P_i) \right| \geq k \right\}$ ;

**Distributie:** Dealerul  $\mathcal{D}$ 

- Transmite participantului  $P_i$  componenta  $Assign(P_i), i \in 1, 2, \dots, n$ ;

**Reconstructie:** Participatii din structura de acces  $\mathcal{A}$ :

- Procedeza identic ca in schema Shamir.

**Fig. 3.** Schema Ito, Saito, si Nishizeki [6]

se in fiecare an. De fiecare data cand cheile sunt inlocuite atunci este necesara recriptarea datelor de pe fiecare baza de date. Cu cat disponibilitatea este mai mare - numarul de noduri duplicate creste- recriptarea lor devine o operatie costisitoare.

Majoritatea tehnicilor de criptarea se bazeaza pe dificultatea factorizarii unui numar sau cea a calcularii logaritmului discret insa o data cu posibila dezvoltare a calculatoarelor cuantice aceste probleme nu vor mai fi atat de dificile. [12]

## 4 Sisteme de stocare de lunga durata bazate pe scheme de partajare

O alternativa la solutia cu criptare care asigura confidentialitatea dar si redundanta necesara este intrebuintarea sistemelor de stocare bazate pe scheme de partajare. [11, 13, 14]

## 4.1 PASIS

În 2000, PASIS este oferit ca o soluție pentru un sistem descentralizat care oferă beneficii precum securitate, redundanță de date și auto-întreținere. Structurile descentralizate împart informația la mai multe noduri folosind scheme de redundanță precum RAID pentru a asigura performanța, scalabilitatea sistemului dar și integritatea datelor. [15]

PASIS folosește schemele de partajare pentru a distribui informația nodurilor de stocare dintr-o rețea. Aceasta presupune folosirea unor agenți pe partea clientului pentru a scrie sau șterge date din noduri. Share-urile obținute dintr-un fișier sau orice obiecte, sunt puse în rețea cu ajutorul agenților. Pe lângă conținutul brut al share-urilor se adaugă și overhead pentru a reține adresa nodului din rețea la care a fost trimisă dar și noul nume cu care este salvată remote.

Considerând o schemă de partajare  $p - m - n$  unde oricare din cei  $m$  participanți pot reconstitui fișierul, dar mai puțin de  $p$  nu obțin vreo informație dintr-un total de  $n$  participanți. Atunci când un candidat inițiază o cerere pentru a citi un fișier atunci agentul PASIS aflat local face următoarele:

- Caută numele celor  $n$  share-uri care alcătuiesc fișierul într-un serviciu care listează toate datele
- Trimite cereri de a citi fișierul la cel puțin  $m$  din cele  $n$  noduri
- În caz că acesta nu primește cel puțin  $m$  răspunsuri continuă pasul anterior încercând query-uri la alte noduri
- Reconstituie fișierul obținut din cele  $m$  share-uri

Operația de scriere este similară cu cea de citire, aceasta oprindu-se atunci când pe cel puțin  $n - m + 1$  noduri s-au scris cu succes share-uri. În articol se menționează și compromisurile de timp/spațiu folosite de PASIS. În schimb, autorii specifică soluții impracticabile pentru auto-mentenanța sistemului, considerând că se poate face prin monitorizare periodică a stării sistemului.

## 4.2 GridSharing

În 2005 Subbiah și Blough propun o nouă abordare pentru a construi un sistem de stocare sigur și tolerant la erori numit GridSharing. Arhitectura este construită pe baza unei scheme de partajare XOR și o multitudine de servere ce folosesc mecanisme de replicare permitându-i schimbarea parametrilor pentru diferite metrice de evaluare a performanțelor. [14]

Articolul identifică 3 tipuri de defectiuni ale serverelor cu ajutorul cărora sunt stocate datele:

- Caderi: un server este *cazut* dacă nu mai răspunde vreunui mesaj din rețea și s-a oprit din a mai efectua vreo operație
- Bizantine: atunci când serverul respectă întotdeauna protocoalele inițiale. Se consideră că share-urile salvate local au fost compromise
- Leakage-only(scăpare?): serverul execută protocoalele corect dar e posibil ca un adversar să fi obținut share-urile stocate

Primele 2 modele definite mai sus sunt preluate din calculul cu sisteme distribuite. Cel de-al 3-lea model a fost introdus pentru a defini atacatorul care foloseste vulnerabilitatile cu intentia doar de a invata din date. Arhitectura Grid-Sharing consta in  $N$  servere unde cel mult  $c$  servere pot cade(? srsly)  $b$  server bizantine si  $l$  leakage only. Cele  $N$  pot fi aranjate intr-un grid cu  $r$  linii si  $N/r$  coloane. Caracteristicile modelului bizantin si leakage-only permit ca share-urile de pe cel mult  $l + b$  linii sa permita dezvaluirea lor unui adeversar.

### 4.3 POTSHARDS

In 2007 este propus un nou sistem care combina caracteristicile PASIS si Grid-Sharing adaugand posibilitatea de a k migrarea datelor la noduri noi. De asemenea este introdusa o tehnica noua de gasire a share-urilor folosind pointeri aproximativi. Pentru a asigura confidentialitatea, autorii adopta o schema de partajare XOR te tipul totul sau nimic, la fel ca in GridSharing. POTSHARDS readreseaza(?) problema in care o persoana neautorizata care incearca sa afle informatii vulnerabile sa nu scape nedetectata. Schemele existente precum PASIS si GridSharing nu indeplineau aceasta cerinta daca un atacator determina locatia share-urilor distribuite.

Solutia pe care o ofera PASIS este reconstruirea share-urilor insa aceasta putea sa aiba repercursiuni negative, precum dezvaluirea unor date secrete, POTSHARDS poate fi gandit ca o aplicatie pe partea de client care comunica cu o multime de noduri (arhive) independente . In prima faza, POTSHARDS partajeaza obiectele in fragmente la care adauga meta-date. Autorii le numesc shard-uri. Shard-urile sunt trimise apoi arhivelor independente, fiecare avand domeniul propriu de securitate. Pentru a reconstitui cu succes obiectele, meta-datele shard-urile contin detalii despre structura pointerilor aproximativi, indicand regiunea in care se afla urmatorul shard.

Pentru un atacator, detinerea unui shard nu il ajuta foarte mult, pentru a detecta urmatorul shard, un atac brut force consta in cereri multiple in zona indicata de pointerul aproximativ. Un astfel de atac nu va trece neobservat de POTSHARDS deoarece unul dintre scopurile sale este sa stocheze datele intr-un mod cat mai "imprastiat". [11]

## 5 Alouneh et al.

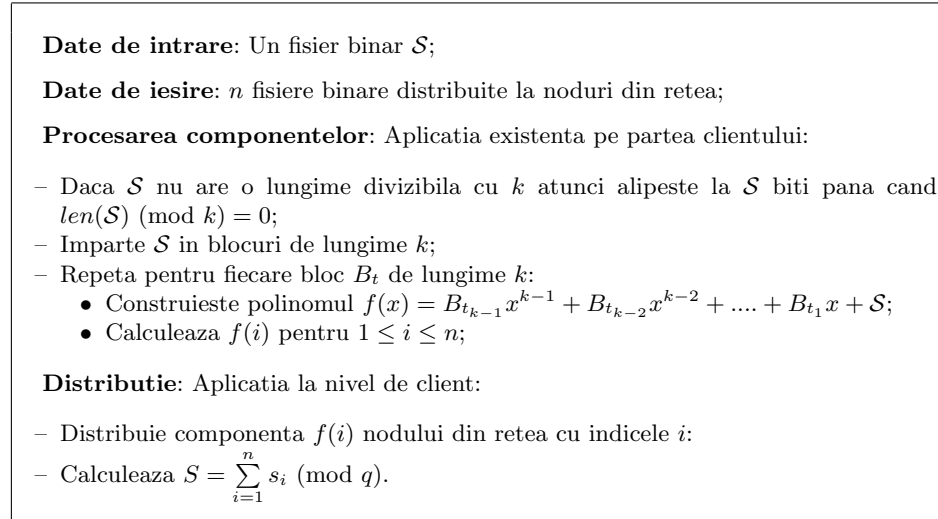
Autorii propun un sistem pentru stocarea datelor un timp indelungat folosind schema Shamir cu cateva modificari. Aceste schimbari se vor arata esentiale mai tarziu in pastrarea securitatii.

### 5.1 Arhitectura sistemului

In cazul in care vrem sa stocam un fisier in sistem (abordand filozofia majoritatii sistemelor de operare - orice este un fisier), acesta este preluat de o aplicatie de control pe partea de client care il imparte in blocuri de biti de lungime  $k$ . Bitii



reprezinta coeficientii unui polinom  $f$ , componenta cu indicele  $i$  va fi reprezentata de valoarea lui  $f(i)$   $i = \{1, 2, \dots, n\}$ . Mentionam ca toate operatiile se vor efectua in  $GF(256)$  modulo un polinom ireductibil (in implementarea sistemului, autorii folosesc  $x^8 + x^5 + x^3 + x + 1$ ). Procedul este descris in detaliu in Fig 4.



**Fig. 4.** Schema Alouneh et al. [16]

Vom exemplifica modul de calcul in  $GF(256) \pmod{g(x)}$  unde  $g(x) = x^8 + x^4 + x^3 + 1$ . Luam polinomul  $f(x) = 10 + 15x$  care corespunde unui fisier format din octetii (in aceasta ordine) 10 15.

$$\begin{aligned}
f(01) \pmod{g(x)} &= 10 + 15 \pmod{g(x)} \\
&= (x^4) + (x^4 + x^2 + 1) \pmod{g(x)} \\
&= x^2 + 1 = 0000\ 00101 \pmod{g(x)} \\
&= 05
\end{aligned} \tag{2}$$

$$\begin{aligned}
f(02) \pmod{g(x)} &= 10 + 15 * 02 \pmod{g(x)} \\
&= (x^4) + (x^5 + x^3 + x) \pmod{g(x)} \\
&= 0011\ 1010 \pmod{g(x)} \\
&= 3A
\end{aligned} \tag{3}$$

Pentru reconstituirea unui fisier se interpoleaza share-urile din orice structura de acces  $\mathcal{A}$  de dimensiune minim  $k$  prin metoda lui Lagrange, asemanator schemei Shamir:

$$f(x) = \sum_{i \in A} f(i) \prod_{j \in A, j \neq i} \frac{x - j}{i - j} \quad (4)$$

## 6 Rezultate obtinute

Impreuna cu mentorul am analizat un articol aparut intr-un jurnal de clasa C unde am indentificat erori majore ale sale si am implementat sistemul descris de autori pentru a demonstra practic, nu doar teoretic anumite greseli pe care le vom evidientia in urmatoarele sectiuni. [16]

### 6.1 Erori gasite in articol

Spre deosebire de schema Shamir in care coeficientii sunt alesi intr-un mod aleator uniform, acestia reprezinta acum continut din fisierele originale. Intuitiv, determinismul implica de cele mai multe ori vulnerabilitati (aplicarea schemei de 2 ori pe date de intrare identice  $\rightarrow$  acelasi date de iesire). Alegerea este motivata de faptul ca multimea share-urilor si efortul computational depus pentru generarea coeficientilor se reduce la un factor de  $k$  spre deosebire de schema Shamir. Natura determinismului duce la cateva atacuri simple in momentul in care un atacator obtine informatiile retinute intr-un nod, indiferent de marimea threshold-ului  $k$ .

### 6.2 Verificarea rezultatelor

### 6.3 Publicarea articolului

## References

1. Katz, J., Lindell, Y.: Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series). Chapman & Hall/CRC (2007)
2. Csirmaz, L.: The size of a share must be large. *Journal of cryptology* **10.4** (1997) 223–231
3. Olimid, R.F.: Secret Sharing-based Group Key Establishment. PhD thesis, University of Bucharest, Romania (2013)
4. Blakley, G.: Safeguarding cryptographic keys. *Proceedings of the 1979 AFIPS National Computer Conference* (1979) 313–317
5. Shamir, A.: How to share a secret. *Commun. ACM* **22**(11) (1979) 612–613
6. Ito, M., Saito, A., Nishizeki, T.: Secret sharing scheme realizing general access structure. *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)* **72**(9) (1989) 56–64
7. Benaloh, J., Leichter, J.: Generalized secret sharing and monotone functions. In: *Proceedings on Advances in Cryptology. CRYPTO '88, New York, NY, USA, Springer-Verlag New York, Inc.* (1990) 27–35
8. Beimel, A.: Secret-sharing schemes: a survey. In: *Coding and cryptology. Springer* (2011) 11–46

9. Hillery, M., Bužek, V., Berthiaume, A.: Quantum secret sharing. *Physical Review A* **59**(3) (1999) 1829
10. Karnin, E.D., Member, S., Greene, J.W., Member, S., Hellman, M.E.: On secret sharing systems. *IEEE Transactions on Information Theory* **29** (1983) 35–41
11. Storer, M.W., Greenan, K.M., Miller, E.L., Voruganti, K.: Potshards - a secure, recoverable, long-term archival storage system. *TOS* **5**(2) (2009)
12. Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, IEEE (1994) 124–134
13. Wylie, J.J., Bigrigg, M.W., Strunk, J.D., Ganger, G.R., Kiliççöte, H., Khosla, P.K.: Survivable information storage systems. *Computer* **33**(8) (2000) 61–68
14. Subbiah, A., Blough, D.M.: An approach for fault tolerant and secure data storage in collaborative work environments. In: *StorageSS*. (2005) 84–93
15. Patterson, D.A., Gibson, G., Katz, R.H.: A case for redundant arrays of inexpensive disks (raid). *SIGMOD Rec.* **17**(3) (June 1988) 109–116
16. Alouneh, S., Abed, S., Mohd, B.J., Kharbutli, M.: An efficient backup technique for database systems based on threshold sharing. *JCP* **8**(11) (2013) 2980–2989