

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. [2] De asemenea, s-au dezvoltat scheme pentru modele de calcul neconventional, cum ar fi cel cuantic. [3]

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 a unor probleme cunoscute). [4]

2 Scheme de partajare

O schemă de partajare constă în distribuirea unei informații secrete \mathcal{S} la mai mulți participanți astfel încât oricare 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 un număr întreg m 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:

- *Inițializare*. Presupune inițializarea variabilelor de mediu necesare.
- *Generare*. O entitate autorizată (numită dealer) \mathcal{D} folosește algoritmul Gen pentru a genera componentele.
- *Distribuire*. 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} [5]:

- *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* [6] în lipsa unor informații oferite doar grupului de acces \mathcal{A} .

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

2.1 Istoric

Primele scheme de partajare au fost dezvoltate independent de Shamir și Blakley în 1979 [7, 8].

Denumite și scheme majoritare (k, n) , acestea rezolvau cazul în care oricare grup de participanți cu un număr mai mare sau egal decât k (mărimea pragului) poate reconstitui secretul \mathcal{S} din componentele primite de la dealer. Dacă schema este perfect *sigură* atunci oricare grup cu un număr de participanți mai mic decât k nu obține vreo informație despre \mathcal{S} .

Notăm $P = \{P_1, \dots, P_n\}$ mulțimea formată din cei n participanți într-o schemă și $y \leftarrow^R Y$ ca y este un element ales uniform aleator din mulțimea Y .

Alte scheme de partajare bazându-se pe structuri de acces generale au fost dezvoltate de Ito, Saito și Nishizeki, realizând o generalizare a schemei Shamir. [9] Benaloh și Leichter au demonstrat ca schemele de partajare de tip prag nu pot fi folosite pe structuri general monotone (familie de submulțimi ale lui P) și obțin o construcție mai eficientă ca Ito et. al [9] din punct de vedere al numărului de componente distribuite participanților. [10]

Autorii prezintă un exemplu trivial pentru care schema Shamir este insuficientă. Considerăm cazul în care vrem să partajăm un secret între 4 participanți: P_1, P_2, P_3, P_4 astfel încât $\{P_1, P_2\}$ 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 unanimă

Presupunând că vrem să împărțim un secret \mathcal{S} la n participanți astfel încât \mathcal{S} să poată fi recuperat doar dacă toți cei n participanți își combină componentele pe care le dețin. Metoda este echivalentă cu o schemă (n, n) majoritară. Un exemplu este schema introdusă de Karin, Greene și Hellman [11] (Fig.)

2.3 Schema Shamir

Schema Shamir oferă mai multă flexibilitate ca schema unanimă prin faptul că oricare k (sau mai mulți) participanți din cei n pot recupera \mathcal{S} , însă mai puțin de k nu obțin nicio informație despre \mathcal{S} . Schema Shamir este deci o schemă (k, n) majoritară.

Intuitiv, având k puncte în plan (x_i, y_i) , $x_i \neq x_j$ $i, j \in \{1, 2, \dots, k\}$ $\forall i \neq j$, există o curbă polinomială unică care trece prin ele. În schimb, pentru a defini o curbă polinomială de grad k care conține $k - 1$ puncte date, există o infinitate de soluții. Evident, orice submulțime de valori s_i de mărime egală cu k este suficientă și necesară pentru a reconstrui polinomul f . După interpolarea componentelor deținute de cel puțin k dintre participanți, secretul \mathcal{S} se va afla în $f(0)$. (Fig. 2)

Pentru un atacator care deține chiar și $k - 1$ valori \mathcal{S}_i , acesta nu determină nimic despre \mathcal{S} , spațiul de soluții posibile fiind identic față de situația în care nu reușește să obțină vreo componentă.

Inițializare:

- Fie $S \in Z_q$ unde $q > 1$ și q prim;
- Fie n numărul de participanți;

Generare: Dealerul \mathcal{D} :

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

Distribuție: Dealerul \mathcal{D}

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

Reconstrucție: Cei n participanți:

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

Fig. 1. Schema unanimă [11]

2.4 Schema Ito, Saito și Nishizeki

În continuare vom descrie modalitatea de distribuire a componentelor de la care au pornit Ito, Saito și Nishizeki pentru ca schema să aibă o structură 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 structură 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 [9]. **TODO prezentarea trebuie să fie de sine stătătoare, trebuie măcar să explici în cuvinte ce înseamnă**

3 Sisteme de stocare de lungă durată

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. [12] Un sistem de stocare necesar nevoilor noastre trebuie să satisfacă cel puțin următoarele 3 condiții:

Inițializare:

- Fie $S \in Z_q$ unde $q > 1$ și q prim;
- Fie n numărul de participanți a.i $q > n$;
- Fie k numărul minim de componente puse în 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 $s_i = f(x_i) \ i \in \{1, 2, \dots, n\}$;

Distribuție: 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 [8]

- Disponibilitatea: Informația trebuie să rămână accesibilă tot timpul, în ciuda erorilor de tip hardware.
- Integritatea: Abilitatea sistemului de a răspunde cererilor într-un mod care garantează corectitudinea lor.
- Confidențialitatea: O persoană care nu face parte din grupul de acces să nu obțină permisiunea de a afla informații de orice fel despre datele existente în sistem.

3.1 Criptare VS scheme de partajare

Una dintre soluțiile existente pentru a construi acest sistem ar putea fi criptarea datelor folosind o cheie înainte de inserarea lor în spațiul de stocare. În momentul în care un user autorizat dorește să efectueze o citire a unor date, întrebuintează cheia potrivită pentru a le decripta. În practică există algoritmi de criptare eficienți precum AES însă aceștia nu garantează confidențialitatea datelor în cazul în care avem de a face cu un adversar fără o limită computațională. Un dezavantaj al criptării este administrarea cheilor, standardele de securitate schimbându-

Inițializare:

- Fie q un număr 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 numărul de participanți;

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 funcția $Assign : P \rightarrow 2^Q$:
 - $Assign(P_i) = D_i \ 1 \leq i \leq n$
 - $\mathcal{A} = \left\{ Q \subseteq Shares : \left| \bigcup_{P_i \in Q} Assign(P_i) \right| \geq k \right\}$;

Distribuție: Dealerul \mathcal{D}

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

Reconstrucție: Participanții din structura de acces \mathcal{A} :

- Procedenza identic ca in schema Shamir.

Fig. 3. Schema Ito, Saito, si Nishizeki [9]

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. [13]

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

O alternativă la soluția cu criptare care asigură confidențialitatea dar și redundanța necesară este întrebuințarea sistemelor de stocare bazate pe scheme de partajare. [12, 14, 15]

4.1 PASIS

PASIS este o soluție pentru un **TODO sistem descentralizat** care oferă beneficii precum securitate, redundanța de date și auto-intreținere. Structurile descentralizate împart informația la mai multe noduri folosind scheme de redundanță precum RAID pentru a asigura performanța, scalabilitatea sistemului dar și integritatea datelor. [16]

PASIS folosește schemele de partajare pentru a distribui informația nodurilor de stocare dintr-o rețea. Aceasta introduce agenți pe partea clientului pentru a scrie sau șterge date din noduri dar și agenți pentru mentenanță. Componentele obținute dintr-un fișier sunt puse în rețea cu ajutorul agenților. Pe lângă conținutul brut al componentelor se adaugă metadate pentru a reține adresa nodului din rețea la care au fost trimise dar și noua denumire cu care este salvată în rețea.

Considerând o schemă de partajare majoritară $p - m - n$ unde oricare din cei m participanți pot reconstitui fișierul, dar mai puțin de p nu obțin nicio informație dintr-un total de n componente. Atunci când un participant inițiază o cerere pentru a citi un fișier, agentul PASIS aflat local procedează după cum urmează:

- Caută numele celor n componente care alcătuiesc fișierul într-un serviciu care listează toate datele.
- Inițiază cereri de citire la cel puțin m din cele n noduri.
- În caz ca acesta nu primește cel puțin m răspunsuri se întoarce la pasul anterior încercând interogări la noduri diferite.
- Reconstituie fișierul obținut din cele m componente.

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 componente. În articol se menționează și compromisurile de timp/spațiu folosite de PASIS. Autorii specifică soluții pentru auto mentenanța sistemului cu ajutorul resurselor umane prin monitorizarea periodică stării sistemului folosind log-uri sau ajustarea parametrilor din cadrul schemei de partajare.

4.2 GridSharing

În 2005 Subbiah și Blough propun o nouă abordare pentru a construi un sistem de stocare securizat și tolerant la erori numit GridSharing. [15]

Schema Shamir nu oferă siguranță în ceea ce privește detectarea sau actualizarea unor componente incorecte introduse de un atacator. Metoda cea mai des folosită este determinarea validității componentelor prin compararea de semnături. Aceasta este realizată prin scheme de verificare non-interactive precum cea a lui Feldman împreună cu schema Shamir [17]

Autorii folosesc un sistem care înlocuiește schemele de verificare cu o schemă de partajare unanimă XOR (considerăm cazul $q = 2$ în Fig. 1) pentru a păstra securitatea construcției. În cazul detectării componentelor incorecte, este adoptată o strategie de tipul **TODO replicate-and-voting**. Componentele sunt **TODO**

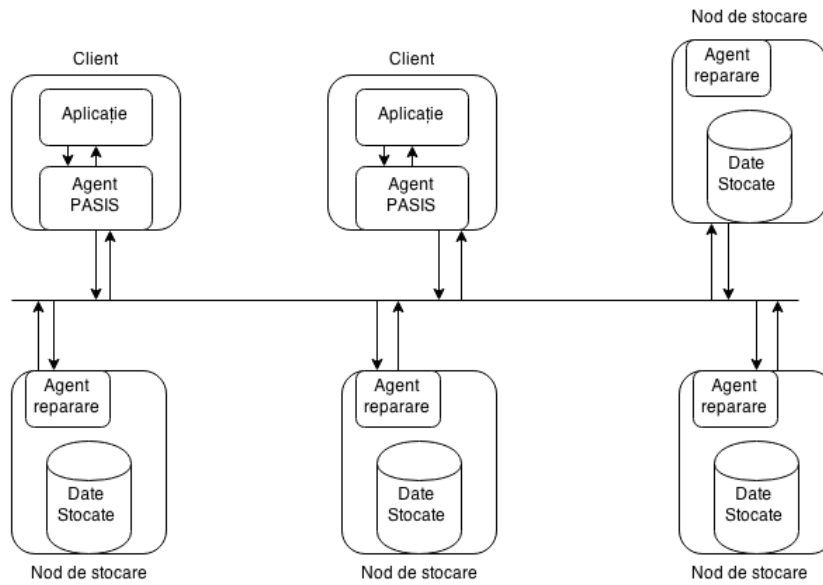


Fig. 4. Arhitectura PASIS cu 4 noduri și 2 clienți

replicate pe un număr mare de servere astfel încât determinarea validității va fi stabilită în funcție de numărul de servere ce le conțin.

Articolul identifică 3 tipuri de defecțiuni care pot apare pe serverele unde sunt stocate datele:

- Abandonări: un server este *abandonat* dacă nu mai raspunde vreunui mesaj din rețea și s-a oprit din a mai efectua vreo operație.
- Bizantine: atunci când serverul respectă întotdeauna protocoalele inițiale dar componentele salvate local au fost compromise.
- Scurgeri de informații: serverul execută protocoalele corect dar e posibil ca un adversar să fi obținut componentele 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 folosește vulnerabilitățile cu intenția de a *învața* din informații.

Arhitectura GridSharing constă în N servere unde cel mult c servere pot fi abandonate, b servere bizantine și l cu scurgeri de informații. Cele N pot fi aranjate într-un grid cu r linii și N/r coloane (considerăm pentru simplitate că $N \pmod{r} = 0$). Caracteristicile modelului bizantin și cel specific scurgerilor de informații permit dezvăluirea componentelor unui adversar de pe cel mult $l + b$ servere.

Example 1. Considerăm ca împartim un secret \mathcal{S} la 3 linii (participanți) astfel încât sistemul să permită 2 componente de tip b , 1 componentă de tip l și 15 servere. În cazul acesta vom folosi o schemă majoritară XOR $((\binom{4}{3}), (\binom{4}{3})) = (3, 3)$.

Vom avea 3 componente, (s_1, s_2, s_3) a.î $s_1 \oplus s_2 \oplus s_3 = \mathcal{S}$. Distribuirea se face în felul următor:

- Serverele situate pe prima linie primesc s_1
- Serverele situate pe a doua linie primesc s_2
- Serverele situate pe a treia linie primesc s_3

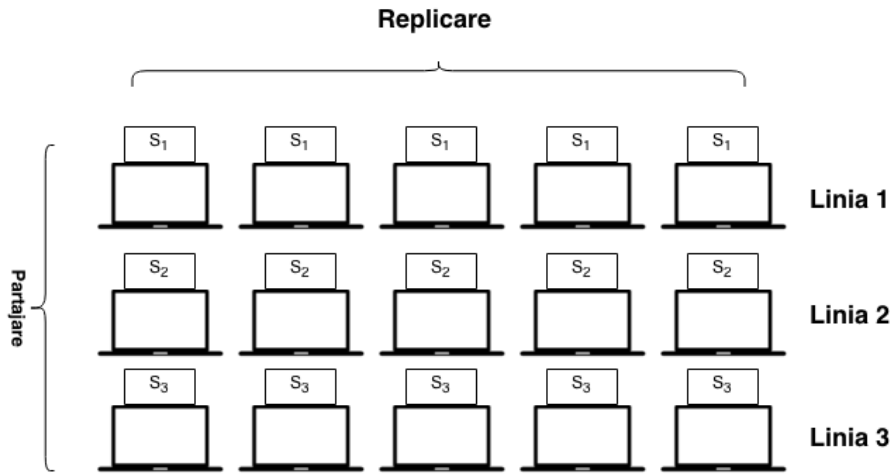


Fig. 5. GridSharing cu 3 linii, 15 servere dintre care 2 bizantine, 1 cu scurgeri de informații

4.3 POTSHARDS

În 2007 este propus un nou sistem care combină caracteristicile PASIS și Grid-Sharing adăugând posibilitatea de migrarea a datelor la noduri noi. De asemenea este introdusă o tehnică nouă de găsim a componentelor folosind pointeri aproximativi. Pentru a asigura confidențialitatea, autorii adoptă o schemă de partajare XOR unanimă, la fel ca în GridSharing. POTSHARDS **TODO readresează** problema în care o persoană neautorizată încearcă să afle informații vulnerabile fără ca aceasta să fie nedetectată. Schemele existente precum PASIS și GridSharing nu îndeplinesc această cerință dacă un atacator determină locația componentelor distribuite inițial.

Soluția pe care o oferă această arhitectură este reconstruirea componentelor într-un mod securizat și folosirea semnăturilor algebrice pentru a asigura un grad ridicat de păstrare a integrității fișierelor. [?] POTSHARDS poate fi gândit ca o aplicație pe partea clientului care comunică cu o mulțime de noduri (arhive) independente.

Ca prim pas, POTSHARDS preprocesează fișierul într-un obiect, partajează obiectul în fragmente la care adaugă meta-date, numite *shards* (Fig. 6) [12]. Acestea sunt trimise apoi arhivelor independente, fiecare având propriul domeniu de securitate, localizate în *regiuni*. Pentru a reconstitui cu succes informația inițială, meta-datele shard-urilor conțin detalii despre structura pointerilor aproximativi, indicând regiunea în care se află următorul shard.

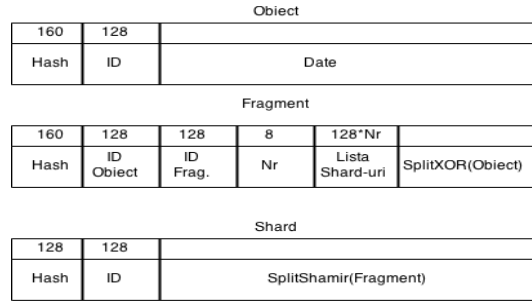


Fig. 6. Entități de date în POTSHARDS. *NR* e numărul de shard-uri produse de un fragment. *SplitXOR* reprezintă o componentă rezultată în urma partajării unanime XOR. Analog *SplitShamir*.

Procesul de fragmentare a datelor este prezentat în Fig. 7.

Pentru ca reconstituirea unui fișier să fie fezabilă unui utilizator, acestuia îi este întoarsă o listă cu locațiile exacte shard-urilor corespunzătoare. Obținerea unui shard de către un atacator, nu este folositoare, pentru a detecta următorul shard, un atac brut force constă în cereri multiple în zona indicată de pointerul aproximativ. Un astfel de atac nu va trece neobservat de POTSHARDS deoarece unul dintre scopurile sale este să stocheze datele într-un mod cât mai uniform distribuit **TODO spread**. [12]

5 Alouneh et al.

Autorii propun un sistem pentru stocarea datelor un timp îndelungat folosind schema Shamir cu câteva modificări. Aceste schimbări se vor arăta cruciale mai târziu în menținerea securității.

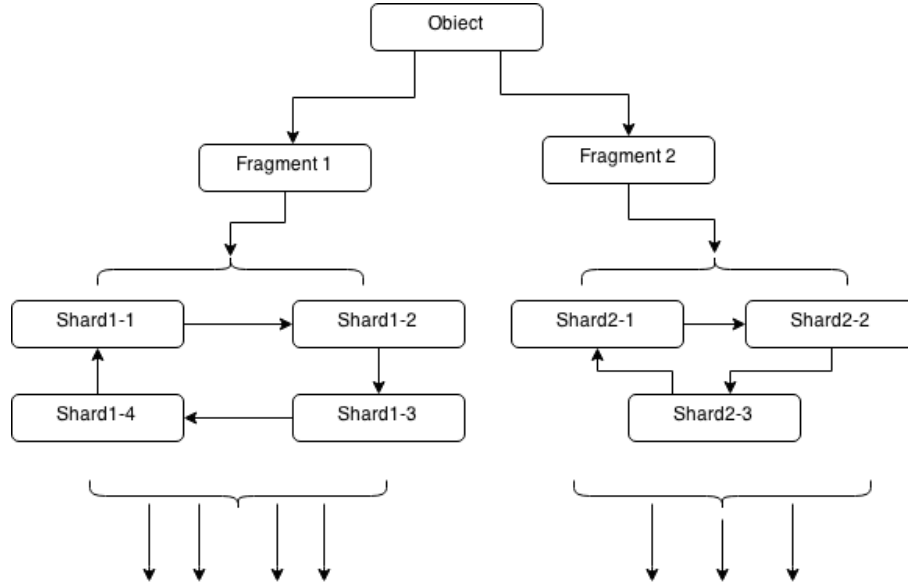


Fig. 7. Distribuirea unui obiect in POTSHARDS

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 imparte in blocuri de octeti de lungime k . Pentru fiecare bloc, octetii devin 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 8.

Example 2. 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.

$$\begin{aligned}
 f(01) \pmod{g(x)} &= 10 + 15 \pmod{g(x)} \\
 &= (x^4) + (x^4 + x^2 + 1) \pmod{g(x)} \\
 &= x^2 + 1 = 000000101_2 \\
 &= 05_{16}
 \end{aligned} \tag{2}$$

Date de intrare: Un fisier binar \mathcal{S} ;

Date de iesire: n fisiere binare distribuite la noduri din retea;

Procesarea componentelor: Aplicatia existenta pe partea clientului:

- Daca \mathcal{S} nu are o lungime divizibila cu k :
 - concateneaza la sfarsitul lui \mathcal{S} octeti pana cand $len(\mathcal{S}) \pmod k = 0$;
- Imparte \mathcal{S} in blocuri de lungime k ;
- Repeta pentru fiecare bloc B_t de lungime k :
 - Construiesc polinomul $f(x) = B_{t_{k-1}}x^{k-1} + B_{t_{k-2}}x^{k-2} + \dots + B_{t_1}x + B_{t_0}$;
 - Calculeaza $f(i)$ pentru $1 \leq i \leq n$;

Distributie: Aplicatia la nivel de client:

- Distribuie componenta $f(i)$ nodului din retea cu indicele i :

Fig. 8. Schema Alouneh et al. - Generare

$$\begin{aligned}
 f(02) \pmod{g(x)} &= 10 + 15 \cdot 02 \pmod{g(x)} \\
 &= (x^4) + (x^5 + x^3 + x) \pmod{g(x)} \\
 &= 00111010_2 \\
 &= 3A_{16}
 \end{aligned} \tag{3}$$

Pentru reconstituirea unui fisier (9) fse interpoleaza din orice multime de componente \mathcal{A} cu dimensiune minim k prin metoda lui Lagrange, asemanator schemei Shamir:

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

Example 3. Vom exemplifica interpolarea 4 pe componentele calculate in 2 si 3 pentru a reconstitui polinomul:

$$\begin{aligned}
 f(x) &= 05(x - 02)(01 - 02)^{-1} + 3A(x - 01)(02 - 01)^{-1} \\
 &= 05(x - 02)03^{-1} + 3A(x - 01)03^{-1} \\
 &= F6(05 + 3A)x + F6(05 \cdot 02 + 3A \cdot 01) \\
 &= F6 \cdot 3F \cdot x + F6 \cdot 30 = 15x + 10
 \end{aligned} \tag{5}$$

Noutatea arhitecturii consta in diminuarea redundantei componentelor la un factor de k , spre deosebire de sistemele descrise in 4.1 sau in 4.3. Reducerea spatiului ocupat este datorat inlocuirii coeficientilor cu octetii din fisierul ce va fi partajat. Confidentialitatea este indusa automat de schema lui Shamir.

Date de intrare: Cel puțin k componente provenite din noduri (distincte);

Date de ieșire: Fisierul binar original S ;

Reconstrucție: Aplicația existentă pe partea clientului:

- Repeta pentru fiecare bloc al lui S :
 - Calculează prin interpolare coeficienții lui $f(x) = B_{t_{k-1}}x^{k-1} + B_{t_{k-2}}x^{k-2} + \dots + B_{t_1} + B_{t_0}$
 - Reconstituie blocul B_t
- Înălțura padding

Fig. 9. Schema Alouneh et al. - Reconstrucție

6 Rezultate obținute

Împreună cu mentorul am analizat un articol apărut într-un jurnal de clasă C unde am identificat erori majore ale sale și am implementat sistemul descris de autori pentru a demonstra practic, nu doar teoretic anumite greseli pe care le vom evidenția în următoarele secțiuni. [18]

6.1 Erori găsite în articol

Spre deosebire de schema Shamir în care coeficienții sunt aleși într-un mod aleator uniform, aceștia reprezintă acum conținut din fișierele originale. Alegerea este motivată de faptul că mulțimea share-urilor și efortul computațional depus pentru generarea coeficienților se reduce la un factor de k spre deosebire de schema Shamir. Natura determinismului duce la câteva atacuri simple în momentul în care un atacator obține informațiile reținute într-un nod, indiferent de mărimea threshold-ului k . Intuitiv, determinismul implică de cele mai multe ori vulnerabilități (aplicarea schemei de 2 ori pe date de intrare identice \rightarrow același date de ieșire).

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. Beimel, A.: Secret-sharing schemes: a survey. In: Coding and cryptology. Springer (2011) 11–46
3. Hillery, M., Bužek, V., Berthiaume, A.: Quantum secret sharing. Physical Review A **59**(3) (1999) 1829

4. Csirmaz, L.: The size of a share must be large. *Journal of cryptology* **10.4** (1997) 223–231
5. Martin, K.M.: Challenging the adversary model in secret sharing schemes. *Coding and Cryptography II, Proceedings of the Royal Flemish Academy of Belgium for Science and the Arts* (2008) 45–63
6. Boneh, D.: The decision diffie-hellman problem. In: *Algorithmic number theory*. Springer (1998) 48–63
7. Blakley, G.: Safeguarding cryptographic keys. *Proceedings of the 1979 AFIPS National Computer Conference* (1979) 313–317
8. Shamir, A.: How to share a secret. *Commun. ACM* **22**(11) (1979) 612–613
9. 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
10. 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
11. 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
12. Storer, M.W., Greenan, K.M., Miller, E.L., Voruganti, K.: Potshards - a secure, recoverable, long-term archival storage system. *TOS* **5**(2) (2009)
13. 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
14. 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
15. Subbiah, A., Blough, D.M.: An approach for fault tolerant and secure data storage in collaborative work environments. In: *StorageSS. (2005)* 84–93
16. 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
17. Feldman, P.: A practical scheme for non-interactive verifiable secret sharing. In: *Proceedings of the 28th Annual Symposium on Foundations of Computer Science. SFCS '87, Washington, DC, USA, IEEE Computer Society* (1987) 427–438
18. 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