TODO

Undergraduate Research Opportunities

Dragoş 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 tehnicililor 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 menţionezi, dar atunci faci asta în secţiunea anterioară, când poţi să adaugi şi alte modalităţi 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 Teoretica a Informatiei

In cazul unor criptosisteme acestea nu pot fi compromise chiar daca adversarul dispune de o putere computationala nelimitata. Cateva exemple de criptosisteme care garanteaza securitatea teoretica-informationala sunt: schemele de partajare, unele protocoale multi-party computation, preluarea intr-un mod sigur(securizat?) informatii de la baze de date. Securitatea teoretica vine insa cu un cost: efortul computational depus este mult mai mare decat in cazul schemelor care nu garanteaza securitatea teoretica (se bazeaza pe dificultatea computationala unor probleme cunoscute). [4]

2 Scheme de partajare

O schemă de partajare constă în distribuirea unei informaţii secrete $\mathcal{S}$ la mai mulţi participanţi $\mathcal{P} = \{P_1, \ldots, P_n\}$ astfel încât oricare mulţime de participanţi predefinită ca făcând parte dintr-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):

- $Gen(S, m)$ este un algoritm care primeşte la intrare un secret S şi un număr întreg m şi întoarce un set de componente $s_1, s_2, \ldots, s_m$.
- $Rec(s_{i1}, s_{i2}, \ldots, s_{iq})$ este un algoritm care primeşte ca parametri de intrare o mulţime de componente şi întoarce S dacă mulţimea $\{P_{i1}, P_{i2}, \ldots, P_{iq}\} \in \mathcal{A}$.

Majoritatea schemelor constau în mai multe etape precum:

- *Iniţializare*. Presupune iniţializarea variabilelor de mediu necesare.
- *Generare*. O entitate autorizată (numită dealer) $\mathcal{D}$ foloseşte algoritmul *Gen* pentru a genera componentele.
- *Distribuţie*. Componentele sunt trimise participanţilor cu ajutorul unui mijloc de comunicare sigur, fără ca acestea sa fie vizibile unui atacator.
- *Reconstrucţie*. Dându-se o mulţime de componente, se foloseşte algoritmul *Rec* pentru a recupera secretul $\mathcal{S}$.

Schemele de partajare se clasifică in funcţie de cantitatea de informaţie secretă pe care o pot obţine persoanele care nu fac parte din $\mathcal{A}$ [5]:

- *Sisteme perfecte de partajare*: componentele nu oferă nici o informatie teoretică despre $\mathcal{S}$ indiferent de resursele computaţionale.
- *Sisteme statistic sigure*: o fracţiune de informaţie este dezvaluită despre $\mathcal{S}$ independent de puterea computională a adversarului.
- *Sisteme computaţional-sigure de partajare*: se bazează pe faptul ca reconstituirea lui $\mathcal{S}$ se reduce la o problema *dificilă* (spre exemplu problema Diffie-Hellman [6]) în lipsa unor informaţii oferite doar grupului de acces $\mathcal{A}$.

În continuare vom prezenta cateva sisteme perfecte de partajare utilizate în cadrul unor arhitecturi pentru stocarea fisierelor pe o durata indelungată.

2.1 Istoric

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

Denumite şi scheme majoritare (k, n), acestea rezolvau cazul în care oricare grup de participanţi 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 participanţi mai mic decat k nu obţine vreo informaţie despre $\mathcal{S}$.

Schemele majoritare (spre exemplu schema Shamir) sunt insuficiente pentru a permite partajarea pentru anumite structuri de acces. Considerăm cazul în care vrem sa partajam un secret între 4 participanţi: P_1, P_2, P_3, P_4 astfel încât $\{P_1, P_2\}$ şi $\{P_3, P_4\}$ să fie singurele mulţimi autorizate pentru reconstrucţia secretului S (i.e. $\mathcal{A} = \{\{P_1, P_2\}, \{P_3, P_4\}\}$). În mod evident, problema nu poate fi rezolvată cu o structură de acces de tip prag: anumite mulţimi de 2 participanţi trebuie să poată reconstrui secretul $(\{P_1, P_2\}, \{P_3, P_4\})$, în timp ce altele nu $(\{P_1, P_3\}, \{P_1, P_4\}, \{P_2, P_3\}, \{P_2, P_4\})$

Astfel de scheme de partajare pentru structuri de acces generale au fost dezvoltate de Ito, Saito şi Nishizeki, realizând o generalizare a schemei Shamir [9]. Benaloh şi Leichter au demonstrat ca schemele de partajare de tip prag nu pot fi folosite pe structuri general monotone (familie de submulţimi ale lui $\mathcal{P}$ cu proprietatea că dacă $A \in \mathcal{A}$ şi $A \subset A'$, atunci $A' \in \mathcal{A}$) şi obţin o construcţie mai eficientă ca Ito et. al din punct de vedere al numărului de componente distribuite participanţilor [10].

2.2 Schema unanimă

Presupunând ca vrem să impărţim un secret $\mathcal{S}$ la n participanţi astfel încât $\mathcal{S}$ sa poată fi recuperat doar daca toţi cei n participanţi îşi combină componentele pe care le deţin. Metoda este echivalentă cu o schemă (n, n) majoritară. Un exemplu este schema introdusă de Karin, Greene şi Hellman (Fig.1) [11].

2.3 Schema Shamir

Schema Shamir oferă mai multă flexibilitate decât schema unanima prin faptul ca oricare k (sau mai multi) participanţi din cei n pot recupera $\mathcal{S}$, însă mai puţin de k participanţi nu obţin nicio informaţie despre $\mathcal{S}$. Schema Shamir este deci o schemă (k, n) majoritară.

Intuitiv, având k puncte in plan (x_i, y_i), $x_i \neq x_j$ $i, j \in \{1, 2, \ldots, k\}$ $\forall i \neq j$, există o curbă polinomială unică care trece prin ele. În schimb, pentru a defini o curbă polinomială de grad k care trece prin $k-1$ puncte date, există o infinitate de soluţii. Evident, orice submulţime de valori s_i de mărime egală cu k este suficientă şi necesară pentru a reconstrui polinomul f. Dupa interpolarea componentelor deţinute de cel puţin k dintre participanţi, secretul $\mathcal{S}$ se determină ca fiind $f(0)$ (Fig. 2) [8].

Iniţializare:

– Fie $S \in Z_q$ unde $q > 1$ şi q prim;
– Fie n numărul de participanţi;

Generare: Dealerul $\mathcal{D}$:

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

Distribuţie: Dealerul $\mathcal{D}$:

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

Reconstrucţie: Cei n participanţi:

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

Fig. 1. Schema unanimă [11]

Pentru un atacator care deţine chiar şi $k-1$ valori s_i, acesta nu determină nimic despre $\mathcal{S}$, spaţiul de soluţii posibile fiind identic faţă de situaţia în care nu reuşeşte sa obţină vreo componentă.

2.4 Schema Ito, Saito şi Nishizeki

În continuare vom descrie modalitatea de distribuire a componentelor de la care au pornit Ito, Saito şi Nishizeki pentru ca schema sa aiba o structură de acces $\mathcal{A} \subseteq 2^P$ (submulţime a setului de participanţi) monotona (i.e. $\forall A \in \mathcal{A}, A \subseteq A' \Rightarrow A' \in \mathcal{A}$). Folosind construcţia unei scheme majoritare (k, n) autorii au reuşit să descrie elementele din $\mathcal{A}$ folosind rezultatul unei reuniuni de mulţimi de componente cu un număr de elemente mai mare sau egal decat k (Fig. 3) [9]. Notaţia $x : Pr$, înseamnă că x are proprietatea Pr.

Dezvantajul acestei structuri este numărul de componente necesar pentru o structură de acces oarecare $\mathcal{A}$. Un mod simplu de construire al funcţiei $Assign$ este Pentru mai multe informatii despre functia $Assign$, cititorul interesat poate citi in [9]. TODO prezentarea trebuie să fie de sine stătătoare, trebuie măcar să explici în cuvinte ce înseamnă

3 Sisteme de stocare de lunga durata

In acesta sectiune vom arata cateva intrebuintari ale schemelor de partajare. Consideram cazul in care vrem sa stocham rapoarte medicale, imagini, documente clasificate pe un timp indelungat intr-un mediu electronic. Pe parcursul

Iniţializare:

- Fie $S \in Z_q$ unde $q > 1$ şi q prim;
- Fie n numărul de participanţi a.i $q > n$;
- Fie k numărul minim de componente puse in comun pentru a determina pe $\mathcal{S}$;

Generare: Dealerul $\mathcal{D}$:

- Alege n valori distincte $x_i \leftarrow^R Z_q$, $i = 1, 2, \ldots, n$;
- Alege $a_i \leftarrow^R Z_q$, $i \in \{1, 2, \ldots, k-1\}$, $a_{k-1} \neq 0$;
- Construieşte polinomul $f(x) = a_{k-1}x^{k-1} + a_{k-2}x^{k-2} + \cdots + a_1 x + \mathcal{S}$;
- Calculează $s_i = f(x_i)$, $i \in \{1, 2, \ldots, n\}$;

Distribuţie: Dealerul $\mathcal{D}$:

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

Reconstrucţie: Orice mulţime cu dimensiunea k (sau mai mare) de participanţi distincţi $P_1, P_2, \ldots, P_k$:

- Interpolează punctele s_i pentru a obţine polinomul f:

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

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

Fig. 2. Schema Shamir [8]

timpului, pot apare in schimb, diverse probleme precum dezastre naturale, defectiunea unor componente hardware, eroare umana, etc. [12] Un sistem de stocare necesar nevoilor noastre trebuie sa satisfaca cel putin urmatoarele 3 conditii:

- 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

Iniţializare:

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

Generare: Dealerul $\mathcal{D}$:

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

Distribuţie: Dealerul $\mathcal{D}$:

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

Reconstrucţie: Participanţii din structura de acces $\mathcal{A}$:

- Procedeaza identic ca in schema Shamir.

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

cheia potrivita pentru a le decripta. In practica exista algoritmi de criptare eficienti precum AES insă aceaştia nu garantează confidenţialitatea 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 schimbanduse 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 lungă durată bazate pe scheme de partajare

O alternativă la soluţia cu criptare care asigură confidenţialitatea dar şi redundanţa necesară este intrebuinţarea sistemelor de stocare bazate pe scheme de partajare. [12, 14, 15]

4.1 PASIS

PASIS este o soluţie pentru un TODO sistem descentralizat care oferă beneficii precum securitate, redundanţa de date si auto-intreţinere. Structurile descentralizate împart informaţia la mai multe noduri folosind scheme de redundanţa precum RAID pentru a asigura performanţa, scalabilitatea sistemului dar şi integritatea datelor. [16]

PASIS foloseşte schemele de partajare pentru a distribui informaţia nodurilor de stocare dintr-o reţea. Aceasta introduce agenţi pe partea clientului pentru a scrie sau şterge date din noduri dar şi agenţi pentru mentenanţă. Componentele obţinute dintr-un fisier sunt puse in reţea cu ajutorul agenţilor4. Pe lângă conţinutul brut al componentelor se adaugă metadate pentru a reţine adresa nodului din reţea la care au fost trimise dar şi noua denumire cu care este salvată in reţea.

Considerănd o schemă de partajare majoritară $p - m - n$ unde oricare din cei m participanţi pot reconstitui fisierul, dar mai puţin de p nu obţin nicio informaţie dintr-un total de n componente. Atunci când un participant iniţiaza o cerere pentru a citi un fisier, agentul PASIS aflat local procedează dupa cum urmează:

- Caută numele celor n componente care alcătuiesc fişierul într-un serviciu care listează toate datele.
- Iniţiaza cereri de citire la cel puţin m din cele n noduri.
- În caz ca acesta nu primeşte cel puţin m răspunsuri se intoarce la pasul anterior încercând interogări la noduri diferite.
- Reconstituie fişierul obţinut din cele m componente.

Operaţia de scriere este similară cu cea de citire, aceasta oprindu-se atunci cand pe cel puţin $n - m + 1$ noduri s-au scris cu succes componente. In articol se menţioneaza şi compromisurile de timp/spatiu folosite de PASIS. Autorii specifică soluţii pentru auto mentenanţa 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 şi Blough propun o nouă abordare pentru a construi un sistem de stocare securizat şi tolerant la erori numit GridSharing. [15]

Schema Shamir nu oferă siguranţă în ceea ce priveşte detectarea sau actualizarea unor componente incorecte introduse de un atacator. Metoda cea

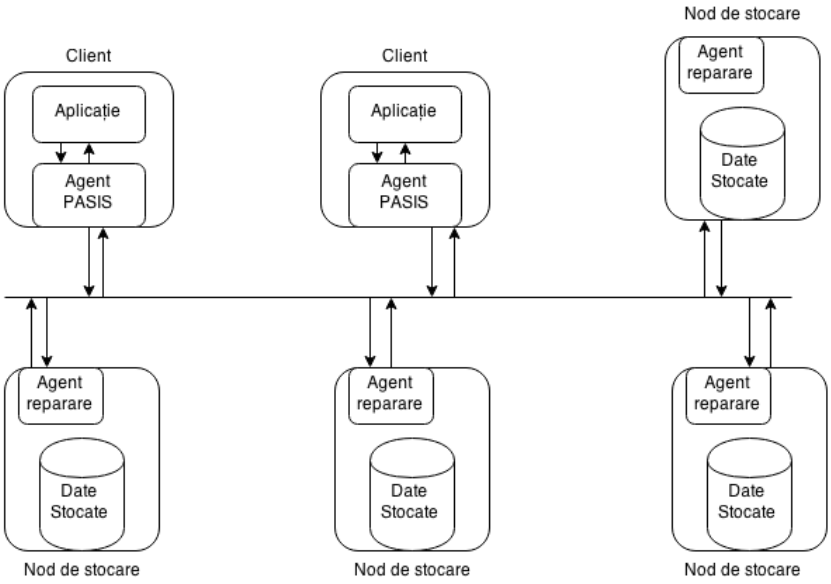

Fig. 4. Arhitectura PASIS cu 4 noduri şi 2 clienţi

mai des folosită este determinarea validităţii 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 înlocuieşte schemele de verificare cu o schemă de partajare unanimă XOR (considerăm cazul $q = 2$ în Fig. 1) pentru a păstra securitatea construcţiei. In cazul detectării componentelor incorecte, este adoptată o strategie de tipul TODO replicate-and-voting. Componentele sunt TODO replicate pe un număr mare de servere astfel încât determinarea validităţii va fi stabilită în funcţie de numărul de servere ce le conţin.

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

- Abandonări: un server este *abandonat* dacă nu mai raspunde vreunui mesaj din retea şi s-a oprit din a mai efectua vreo operaţie.
- Bizantine: atunci când serverul respectă întotdeauna protocoalele iniţiale dar componentele salvate local au fost compromise.
- Scurgeri de informaţii: serverul execută protocoalele corect dar e posibil ca un adversar să fi obţinut 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 foloseşte vulnerabilităţile cu intenţia de a *învaţa* din informaţii.

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

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

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 in 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

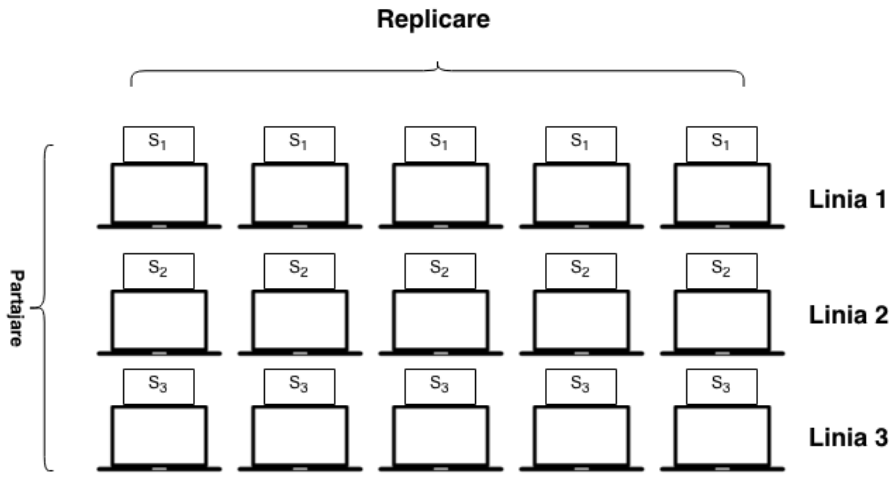

Fig. 5. GridSharing cu 3 linii, 15 servere dintre care 2 bizantine, 1 cu scurgeri de informaţii

4.3 POTSHARDS

În 2007 este propus un nou sistem care combină caracteristicile PASIS şi GridSharing adăugând posiblitatea de migrarea a datelor la noduri noi. De asemenea

este introdusă o tehnică nouă de găsire a componentelor folosind pointeri aproximativi. Pentru a asigura confidenţialitatea, autorii adoptă o schemă de partajare XOR unanimă, la fel ca în GridSharing. POTSHARDS TODO readresează problema în care o persoană neautorizată încearca să afle informaţii vulnerabile fără ca aceasta sa fie nedetectată. Schemele existente precum PASIS şi GridSharing nu indeplineau această cerinţa dacă un atacator determină locaţia componentelor distribuite iniţial.

Soluţia pe care o oferă această arhitectură este reconstruirea componentelor într-un mod securizat si folosirea semnăturilor algebrice pentru a asigura un grad ridicat de păstrare a integritaţii fişierelor. [18] POTSHARDS poate fi gândit ca o aplicaţie pe partea clientului care comunică cu o mulţime de noduri (arhive) independente.

Ca prim pas, POTSHARDS preprocesează fişierul 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 in *regiuni*. Pentru a reconstitui cu succes informaţia iniţiala, meta-datele shard-urilor conţin detalii despre structura pointerilor aproximativi, indicând regiunea în care se află următorul shard.

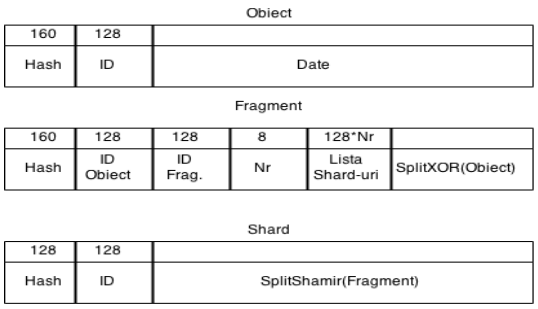

Fig. 6. Entităţi de date in POTSHARDS. NR e numărul de shard-uri produse de un fragment. $SplitXOR$ reprezintă o componentă rezultată în urma partajarii unanime XOR. Analog $SplitShamir$.

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

Pentru ca reconstituirea unui fişier sa fie fezabilă unui utilizator, acestuia îi este întoarsă o listă cu locaţiile exacte shard-urilor corespunzătoare. Obţinerea 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 deorece unul dintre scopurile sale este să stocheze datele într-un mod cat mai uniform distribuit TODO spread. [12]

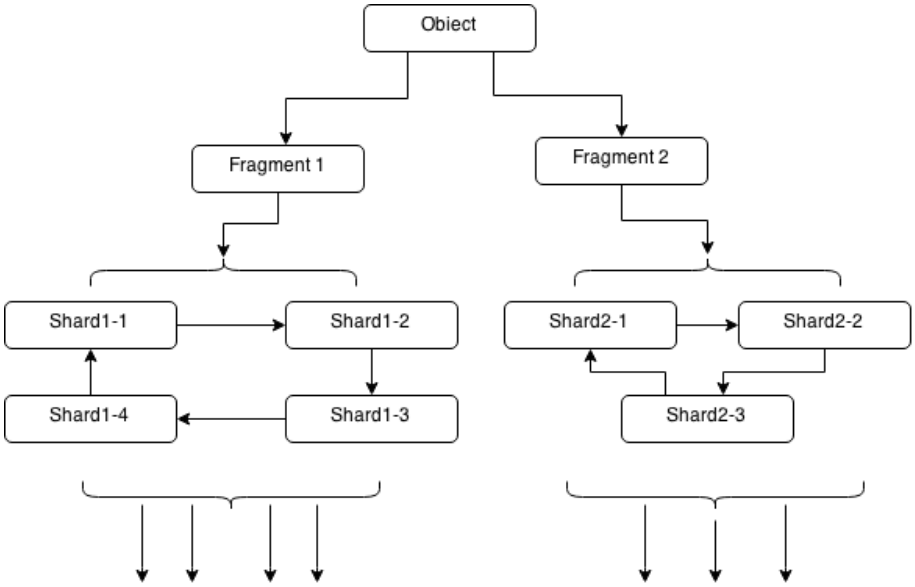

Fig. 7. Distribuirea unui obiect in POTSHARDS

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 cruciale mai tarziu in mentinerea 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 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 $f(i)$ $i = \{1, 2, \ldots, n\}$. Mentionam ca toate operatiile se vor efectua in $GF(256)$ modulo un polinom ireductibil (in implementarea sistemului, autorii folosesc $x^8 + x^5 + x^3 + x + 1$). Procedeul este descris in detaliu in Fig 8.

Example 2. Vom exemplifica modul de calcul in $GF(256)$ (mod $g(x)$) unde $g(x) = x^8 + x^4 + x^3 + 1$. Luam polinomul $f(x) = 10 + 15x$ care corespunde

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:
  - Construieste polinomul $f(x) = B_{t_{k-1}} x^{k-1} + B_{t_{k-2}} x^{k-2} + ... + B_{t_1} x + B_{t_0}$;
  - 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

unui fisier format din octetii (in aceasta ordine) 10 15.

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

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

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:

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

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 putin k componente provenite din noduri (distincte);

Date de iesire: Fisierul binar original $\mathcal{S}$;

Reconstructie: Aplicatia existenta pe partea clientului:

- Repeta pentru fiecare bloc al lui $\mathcal{S}$:
  - Calculeaza prin interpolare coeficientii lui $f(x) = B_{t_{k-1}}x^{k-1} + B_{t_{k-2}}x^{k-2} + ... + B_{t_1} + B_{t_0}$
  - Reconstituie blocul B_t
- Inlatura padding

Fig. 9. Schema Alouneh et al. - Reconstructie

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 evidentia in urmatoarele sectiuni. [19]

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. 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 treshold-ului k. Intuitiv, determinismul implica de cele mai multe ori vulnerabilitati (aplicarea schemei de 2 ori pe date de intrare identice $\rightarrow$ acelasi date de iesire).

6.2 Verificarea rezultatelor

6.3 Publicarea articolului

Referinţe

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. Schwarz, T.J.E., Miller, E.L.: Store, forget, and check: Using algebraic signatures to check remotely administered storage. In: 26th IEEE International Conference on Distributed Computing Systems (ICDCS 2006), 4-7 July 2006, Lisboa, Portugal. (2006) 12
19. 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