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

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). [2]

2 Scheme de partajare

O schema de partajare consta in distribuirea unui obiect, o informatie secreta $\mathcal{S}$ la mai multi participanti astfel incat oricare grup predefinit inainte (o structura de acces pe care o vom denumi $\mathcal{A}$) sa poate reconstitui secretul $\mathcal{S}$. Mai formal o schema de partajare este reprezentata de o pereche de algoritmi (Gen, Rec):

- $Gen(S, m)$ este un algoritm care primeste la intrare un secret S si are ca date de iesire un set de componente $s_1, s_2, \ldots, s_m$.
- $Rec(s_{i1}, s_{i2}, \ldots, s_{iq})$ este un algoritm care ia ca parametri de intrare o multime de componente si are ca date iesire S daca multimea $s_{i1}, s_{i2}, \ldots, s_{iq}$ este *autorizata*. [3]

Majoritatea schemelor au la baza mai multe etape precum:

- *Initializare*. Presupune initializarea variabilelor de mediu necesare.
- *Generare*. O terta autorizata (dealer) $\mathcal{D}$ foloseste algoritmul *Gen* pentru a genera share-urile.
- *Distributie*. Componentele sunt trimise participantilor cu ajutorul unui canal sigur, fara ca acestea sa fie vizibile unui atacator.
- *Reconstructie*. Dandu-se o multime de share-uri, se foloseste algoritmul *Rec* pentru a recupera secretul $\mathcal{S}$

Acestea se clasifica in functie de cantitatea de informatie secreta pe care o pot obtine persoanele care nu fac parte din $\mathcal{A}$:

- *Sisteme perfecte de partajare*: componentele nu ofera nici o informatie teoretica despre $\mathcal{S}$
- *Sisteme computational-sigure de partajare*: se bazeaza pe faptul ca reconstituirea lui $\mathcal{S}$ se reduce la o problema *NP-completa* in lipsa unor informatii oferite doar grupului de acces $\mathcal{A}$

In continuare vom prezenta cateva sisteme perfecte de partajare pentru a forma bazele unor arhitecturi pentru stocarea fisierelor pentru o durata indelungata.

2.1 Constructii existente

Primele scheme de partajare au fost dezvoltate independent de Adi Shamir si George 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 (dimensiunea thresholdului) 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 $P := P_1, P_2, \ldots, P_n$ 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 bazandu-se pe grupuri speciale de acces au fost dezvoltate de Ito, Saito, si Nishizeki, realizand o generalizare a schemei lui Shamir. [6]

Benaloh si Leichter au demonstrat ca schemele de partajare threshold nu pot garanta decat construirea unei fractiuni din multimea functiilor de partajare [7].

Autorii prezinta un exemplu trivial pentru care schema lui Shamir este insuficienta. Consideram cazul in care vrem sa partajam un secret unor 4 participanti: P_1, P_2, P_3, P_4 astfel incat $P_1 + P_2 = \mathcal{S}$ si $P_3 + P_4 = \mathcal{S}$, unde cu + notam operatia de reuniune a componentelor primite de 2 persoane. Vrem ca celelalte combinatii de share-uri detinute sa nu poate reconstitui $\mathcal{S}$.

Dezavantajul acestor scheme este dimensiunea share-urilor, facand adesea majoritatea constructiilor impracticabile. [8] De asemenea, s-au dezvoltat scheme pentru modele de calcul neconventional, cum ar fi cel cuantic. [9]

2.2 Schema unanima

Presupunand ca vrem sa impartim un fisier $\mathcal{S}$ la n participanti fara ca $\mathcal{S}$ sa poate fi recuperat doar daca cei n isi combina share-urile detinute. Metoda este echivalenta cu o schema (n, n) majoritara. Un exemplu este schema introdusa de Karin, Greene si Hellman [10]

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

Distributie: Dealerul $\mathcal{D}$

- trimite participantului P_i componenta s_i, $i \in 1, 2, \ldots, n$;

Reconstructie: Cei n participanti:

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

Fig. 1. Schema unanima [10]

2.3 Schema lui Shamir

Schema lui 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 putin de k nu obtin nicio informatie despre $\mathcal{S}$.

Intuitiv, avand m puncte in plan (x_i, y_i), $x_i \neq x_j$ $i, j = 1, 2, \ldots, m$ $or.i \neq j$, exista o curba polinomiala unica ce trece prin ele. In schimb, pentru a defini o curba polinomiala de grad m care contine $m-1$ 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 putin k dintre participanti, secretul $\mathcal{S}$ se va afla in $f(0)$.

Pentru un atacator care detine chiar si $k-1$ valori $\mathcal{S}_i$, acesta nu invata nimic despre $\mathcal{S}$, spatiul de solutii posibile fiind identic fata de situatia in care nu reuseste sa obtina vreo componenta.

Initializare:

- Fie $S \in Z_q$ unde $q > 1$ si q prim;
- Fie n numarul de participanti a.i $q > n$;
- Fie k dimensiunea majoritatii.

Generare: Dealerul $\mathcal{D}$:

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

Distributie: Dealerul $\mathcal{D}$:

- Trimite participantului P_i componenta s_i, $i \in 1, n-1$;

Reconstructie: Orice multime de k participanti distincti $P_1, P_2, \ldots, P_k$:

- interpoleaza punctele s_i - pentru simplitate vom considera metoda lui Lagrange - pentru a obtine polinomul f:

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

- afla secretul reconstruit $S = f(0)$;

Fig. 2. Schema lui Shamir [5]

2.4 Schema Ito, Saito, si Nishizeki

In continuare vom descrie modalitatea de distribuire a share-urilor de la care au pornit Ito, Saito si Nishizeki pentru ca schema sa aiba o structura de acces $\mathcal{A} \subseteq 2^P$. Autorii au pornit de la constructia unei scheme majoritare (k, n) pentru a putea descrie elementele lui $\mathcal{A}$ ca fiind rezultatul unei reuniuni de multimi de share-uri cu un numar de elemente mai mare sau egal decat k. Explicatii mai detaliate despre constructia aceasta se afla in Fig. 3.

Dezvantajul acestei structuri este numarul de componente per structura $\mathcal{A}$. Pentru mai multe informatii despre functia $Assign$, cititorul interesat poate citi in [6].

Initializare:

- Fie q un numar prim q, $q > 1$ si $z \in \mathbb{N}$ nenul. Luam corpul $\mathcal{C} = GF(p^z)$;
- 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 $x_i \leftarrow^R Z_q$, $i = 1, 2, \ldots, n$;
- Alege $a_i \leftarrow^R \mathcal{C} - 0$, $i = 1, 2, \ldots, k-1$, $a_{k-1} \neq 0$;
- Construieste polinomul $f(x) = a_{k-1} * x^{k-1} + a_{k-2} * x^{k-2} + ... + a_1 * x + \mathcal{S}$;
- Atribuie $s_i := f(x_i)$ $i = 1, 2, \ldots, n$; Notam aceasta multime cu $Shares$;
- Alege $D_i \subseteq Shares$ $1 \leq i \leq n$;
- Alege functia $Assign : P \rightarrow 2^Q$:
  - $Assign(P_i) = D_i$ $1 \leq i \leq n$
  - $\mathcal{A} = \{ Q \subseteq Shares : |\bigcup_{i \in Q} Assign(i)| \geq k\}$;

Distributie: Dealerul $\mathcal{D}$

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

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

- Procedeaza identic ca in schema lui Shamir;

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

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 timpului, pot apare in schimb, diverse probleme precum dezastre naturale, defectiunea unor componente hardware, eroare umana, etc. [11] 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 cheia potrivita pentru a le decripta. In practica exista algoritmi de criptare eficienti precum AES insa aceastia nu garanteaza confidentialitatea datelor in cazul in 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. [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

In 2000, PASIS este oferit ca o solutie pentru un sistem descentralizat care ofera beneficii precum securitate, redundanta de date si auto-intretinere. Structurile

descentralizate impart informatia la mai multe noduri folosind scheme de redundanta precum RAID pentru a asigura performanta, scalabilitatea sistemului dar si integritatea datelor. [15]

PASIS foloseste schemele de partajare pentru a distribui informatia nodurilor de stocare dintr-o retea. Aceasta presupune folosirea unor agenti pe partea clientului pentru a scrie sau sterge date din noduri. Share-urile obtinute dintr-un fisier sau orice obiecte, sunt puse in retea cu ajutorul agentilor. Pe langa continutul brut al share-urilor se adauga si overhead pentru a retine adresa nodului din retea la care a fost trimisa dar si noul nume cu care este salvata remote.

Considerand o schema de partajare $p-m-n$ unde oricare din cei m participanti pot reconstitui fisierul, dar mai putin de p nu obtin vreo informatie dintr-un total de n participanti. Atunci cand un candidat initiaza o cerere pentru a citi un fisier atunci agentul PASIS aflat local face urmatoarele:

- Cauta numele celor n share-uri care alcatuiesc fisierul intr-un serviciu care listeaza toate datele
- Trimite cereri de a citi fisierul la cel putin m din cele n noduri
- In caz ca acesta nu primeste cel putin m raspunsuri continua pasul anterior incercand query-uri la alte noduri
- Reconstituie fisierul obtinut din cele m share-uri

Operatia de scriere este similara cu cea de citire, aceasta oprindu-se atunci cand pe cel putin $n - m + 1$ noduri s-au scris cu succes share-uri. In articol se mentioneaza si compromisurile de timp/spatiu folosite de PASIS. In schimb, autorii specifica solutii impracticabile pentru auto mentenanta sistemului, considerand ca se poate face prin monitorizare periodica a starii sistemului.

4.2 GridSharing

In 2005 Subbiah si Blough propun o noua abordare pentru a construi un sistem de stocare sigur si tolerant la erori numit GridSharing. Arhitectura este construita pe baza unei scheme de partajare XOR si o multitudine de servere ce folosesc mecanisme de replicare permitandu-i schimbarea parametrilor pentru diferite metrici de evaluare a performantelor. [14]

Articolul identifica 3 tipuri de defectiuni ale serverelor cu ajutorul carora sunt stocate datele:

- Caderi: un server este *cazut* daca nu mai raspunde vreunui mesaj din retea si s-a oprit din a mai efectua vreo operatie
- Bizantine: atunci cand serverul respecta intotdeauna protocoalele initiale. Se considera ca share-urile salvate local au fost compromise
- Leakage-only(scapare?): serverul executa protocoalele corect dar e posibil ca un adversar sa fi obtinut 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 GridSharing 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 GridSharing adaugand posiblitatea 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 deorece unul dintre scopurile sale este sa stocheze datele intr-un mod cat mai "imprastiat". [11]

5 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. [16]

5.1 Arhitectura sistemului

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

In cazul in care vrem sa stocam un fisier in sistem (abordand filozofia majoritatii sistemelor de operare - orice este un fisier), este preluat de o aplicatie

de control care il imparte in blocuri de biti de lungime k. Bitii reprezinta coeficientii unui polinom f, share-ul cu index i va fi reprezentat de valoarea lui $f(i)i = 1, 2, \ldots, n$. Mentionam ca toate operatiile se vor efectua in $GF(256)$ modulo un polinom ireductibil. Autorii folosesc $x^8 + x^5 + x^3 + x + 1$.

Pentru reconstituirea unui fisier se interpoleaza share-urile din orice multime A de dimensiune minim k prin metoda lui Lagrange, asemanator schemei lui Shamir:

$$f(x) = \sum_{i \in A} f(i) \prod_{j \in A, j \neq i} \frac{x - j}{i - j} \tag{2}$$

5.2 Erori gasite in articol

Spre deosebire de schema lui 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 treshold-ului k.

5.3 Verificarea rezultatelor

5.4 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