Rassu Michael
Monetta Quentin
Baumel Baptiste
Islahi Chaimae
Bouzid Fares

M1 SICOM

Stockage de données protégées avec schéma de shamir

1. Introduction :

Le partage de clé secrète de Shamir (Shamir's Secret Sharing) est un algorithme de cryptographie. C'est une forme de partage de secret, où un secret est divisé en parties, donnant à chaque participant sa propre clé partagée, où certaines des parties ou l'ensemble d'entre elles sont nécessaires afin de reconstruire le secret.

L'objectif est d'implémenter un code en Python qui divise et stocke des données avec un schéma de Shamir.

2. Définition mathématique :

Notre objectif est de diviser certaines données D en N pièces (D_1, D_2 , D_3 … D_n telle sorte que :

1. la connaissance de k-1 ou plus D_i pièces rend D facilement calculable.
2. la connaissance de k ou moins D_i pièces rend D complètement indéterminée.

3. Système de partage de secret de Shamir :

L'idée est que :

- 2 points sont suffisants pour définir une ligne.
- 3 points pour définir une parabole.
- 4 points pour une courbe cubique ...

Donc il faut k points pour définir un polynôme de degré k-1.

on choisit au hasard (k-1) coefficients : a_1 , a_2 ,a_3 ... a_{k-1}

et on pose a_0 = S.

On construit le polynôme $f(x) =$ a0 + a_1x + a_2x^2 + …+ a $_{k-1}$ x $^{k-1}$

Donc, étant donné un sous-ensemble de k de couples, nous pouvons trouver les coefficients du polynôme à l'aide de l'interpolation polynomiale, le secret S étant le terme constant a_0.

4. Propriétés :

Certaines des propriétés utiles du schéma seuil (k,n) de Shamir sont les suivantes :

- Sécurisé : la sécurité de l'information est théorique
- Minimal : La taille de chaque pièce ne dépasse pas la taille des données d'origine.
- Extensible : Quand k est fixe, D_i morceaux peuvent être ajoutés ou supprimés de manière dynamique sans affecter les autres morceaux.
- Dynamique : La sécurité peut être facilement améliorée sans changer le secret, mais en changeant le polynôme de temps en temps (en gardant le même premier terme du polynôme a_0) et en construisant alors les nouvelles parties pour les participants.
- Flexible : Dans les organisations où la hiérarchie est importante, nous pouvons fournir à chaque participant un nombre différent de pièces en fonction de son importance dans l'organisation.

5. Fonctionnement :

Pour comprendre l'algorithme, commençons par une situation très simple., Alice et Bob, voudrait partager le code à 4 chiffres de leur carte bancaire, un nombre secret noté S compris entre 0000 et 9999.

Pour cela, il suffit de choisir deux nombres entiers a et b tels que a+b=S et de fournir a à Alice et b à Bob. S'ils veulent faire un achat commun, il leur suffit d'ajouter leurs codes respectifs.

Tout ceci est très simple, mais il y a une faille potentielle : supposons qu'Alice se voie attribuer le code a=9958. Elle dispose alors d'une information capitale sur S : il est compris entre 9958 et 9999, elle peut donc le trouver en au plus 42 tentatives.

→ Pas très robuste

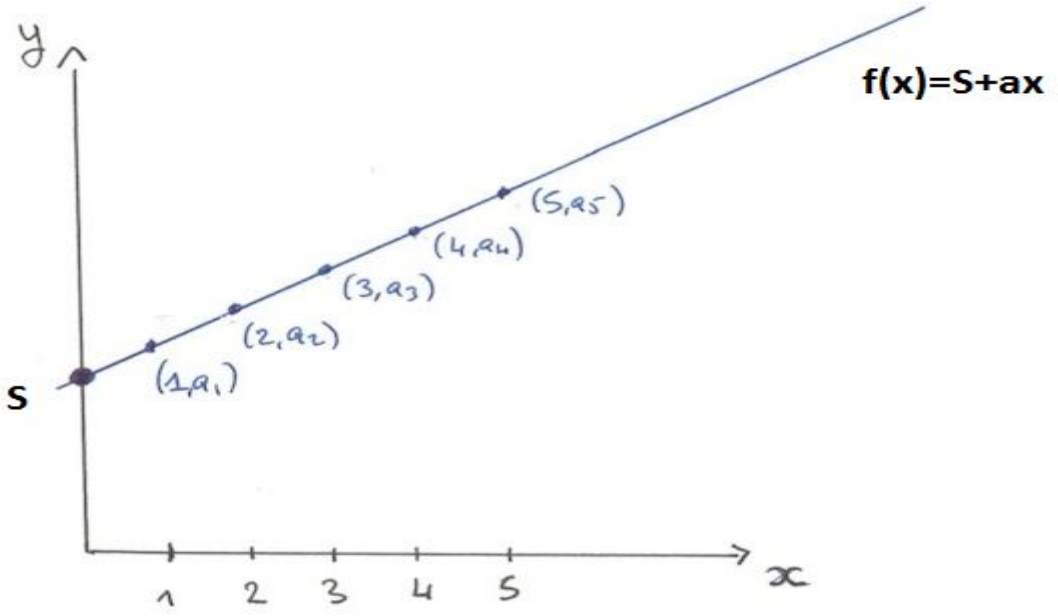
Pour y remédier, une astuce classique est de travailler modulo 10000, c'est à dire de calculer en ajoutant ou retirant autant de multiples de 10000 que nécessaire pour revenir entre 0000 et 9999, par exemple 8545+2651=11196=1196. On peut alors choisir a au hasard entre 0000 et 9999 et poser **b=S−a mod(10000)**.

On voit que maintenant la connaissance de a n'apporte plus aucune information sur S.

Comment faire maintenant pour partager un secret à n≥3 personnes de sorte que 2 personnes au moins soient nécessaires pour le révéler ?

L'idée de Shamir est de tout simplement considérer....une droite. En effet, une droite est complètement déterminée par deux de ses points, et à l'inverse la connaissance d'un point sur cette droite ne révèle que peu d'information sur celle-ci. En formules, supposons que notre secret est un entier naturel S, et choisissons une droite dans le plan passant par le point (0,S), son équation est donc de la forme **y=mx+S**.

L'information donnée au participant A_i est le point d'abscisse i sur la droite, soit le point de coordonnées **(i,a_i)**. Si deux participants entrent leurs codes (i,a_i) et (j,a_j), l'ordinateur calcule la pente et on déduit S.



6. Utilisation par l'exemple :

a- Variables

S = Secret (S1 + S2)

n = Secret à partager en n fois

k = il faut k points pour définir un polynôme de degré k−1.

b- Etapes

1) Il faut choisir un schéma de seuil (k, n) pour partager le secret S
2) Ce qui revient donc à définir k-1 coefficients tel que : $a_1, a_2 \ldots, a_{k-1}$
3) Construire le polynôme $f(x) = S + a_1 x + a_2 x^2$

c- Exemple

S = (1234) n = 6 k = 3

L'algorithme nous génère 2 nombres : (a_1 = 166) ; (a_2 = 94)

On construit le polynôme grâce à ses valeurs :

$f(x) = 1234 + 166x + 94\ x^2$

Avec n = 6, on réalise 6 fois la fonction f(x) pour obtenir les couples de valeurs suivants :

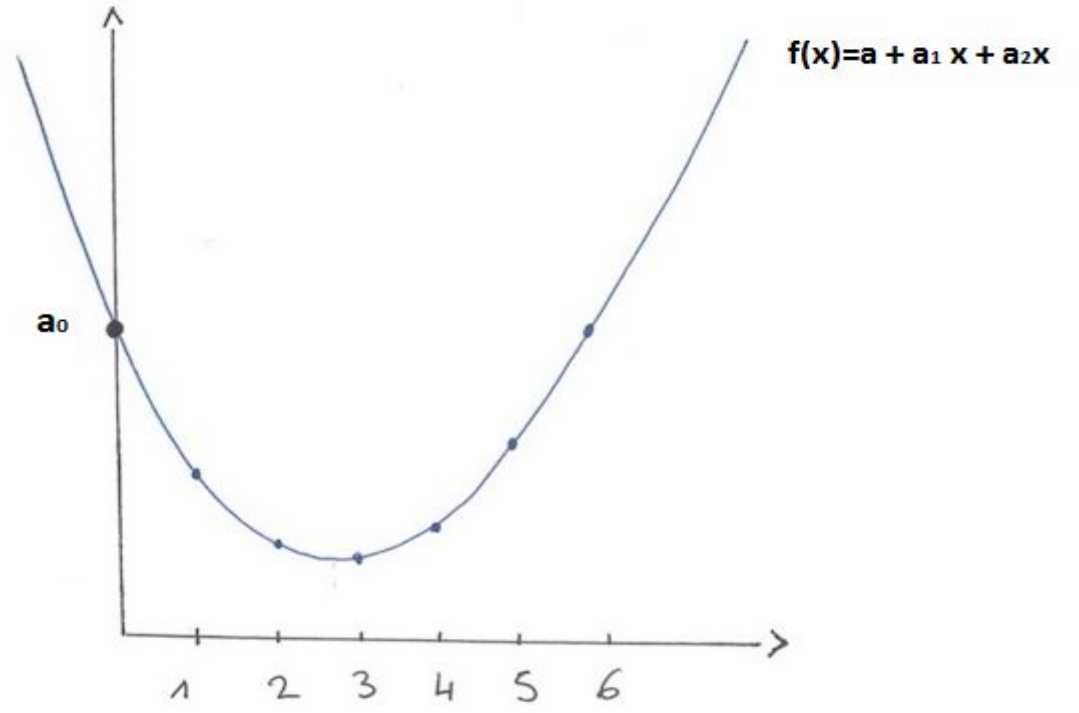
f(1) = 1494 f(2) = 1942 f(3) = 2578 f(4) = 3402 f(5) = 4414 f(6) = 5614

Chaque personne recevra un x et un f(x).

7. La reconstitution

Pour reconstituer un polynôme de degré 2, il nous faut au moins k =2. Ainsi, il faudrait un polynôme de la forme :

$f(x) = a_0 + a_1 x + a_2 x^2$



Exemple :

Le secret a été partagé 4 fois de la manière suivante :

f(5) = 3 f(7) = 2 f(12) = 6 f(30) = 15

Il n'y a qu'un seul polynôme de degré 3 qui passe par ses 4 points. Ainsi, l'objectif est de retrouver le polynôme. Pour cela, on utilise Le polynôme de Lagrange qui d'interpoler une série de points par un polynôme qui passe par ces noeuds.

$$g(0) = \sum_{i=1}^{k} D_i \left(\prod_{j=1, j \neq i}^{k} \frac{-x_j}{x_i - x_j} \right)$$

$$D \equiv g(0) \mod p$$

Pour calculer la reconstitution, nous avons besoin de k qui est le minimum de points pour définir le polynômes.

→ Pour k points $(x_i, ; y_i)$: i = i,...,k

→ On calcule grâce au polynôme de Lagrange g(0)

→ Ensuite, nous devons réaliser le modulo de ce résultat avec une valeur p, respectant les conditions suivantes :

- Ne doit pas être négatif ou nul.
- Doit être supérieur à S et à n.
- Doit être premier .

Le résultat de ce calcul représente notre secret

$$\mathbf{g(0) \bmod p = S.}$$

Pour retrouver les fragments des secrets, on doit tout d'abord générer un polynôme tel que :

$$\mathbf{q(x) = S + a_1x + a_2x^2}$$

où a_1 et a_2 sont deux nombres choisi de manière aléatoire.

Ainsi, on obtient :

→ **q(1) modulo p** = premier fragment du secret

→ **q(2) modulo p** = deuxième fragment du secret

.

..

→ **q(k-1) modulo p** = dernier fragment di secret

Implémentation :

Dans cette partie, nous avons implémenté un code en Python pour diviser et stocker une donnée avec le schéma de Shamir.

→ Voir le code StockageShamir.py

Sources :

https://fr.wikipedia.org/wiki/Partage_de_cl%C3%A9_secr%C3%A8te_de_Shamir

http://images.math.cnrs.fr/Comment-partager-un-secret.html

https://ljk.imag.fr/membres/Bernard.Ycart/mel/ev/node21.html

http://www.cs.haifa.ac.il/~orrd/IntroToCrypto/Spring11/Lecture11.pdf

https://team.inria.fr/diana/team-members/damien-saucez/