Rapport du projet de cryptographie

Modélisation Mathématiques

Mathis Deloge, Antoine Petot, Ange Picard

12 décembre 2016

1 Descriptif du sujet

Nous étudierons ici deux algorithmes bien connus en cryptographie, l'échange de clé de Diffie-Hellman et le chiffrement par transposition. Pour illustrer nos propos, nous ferons appel à Alice et Bob, deux personnages voulant s'échanger une clé de cryptage afin de s'échanger des messages sécurisés sans que personne ne puisse les intercepter. Nous les conseillerons tout d'abord sur le choix du protocole leur permettant de partager une clé de chiffrement puis nous argumenterons l'utilisation de chacun des protocoles, les modèles mathématiques sur lesquels ils se basent, ainsi que leurs failles.

Différents prolongements nous permettrons d'étudier plus précisément ces protocoles de cryptographie, notamment le problème du logarithme discret ainsi que l'algorithme "baby step giant step" permettant de s'approcher de la résolution de ce problème.

1.1 Diffie-Hellman

Diffie-Hellman est un protocole d'échange de clé créé par Whitfield Diffie et Martin Hellman après leur découverte du principe de la cryptographie à clé publique. Il permet notamment à deux personnes (ici, Alice et Bob) de s'échanger une clé de chiffrement (un nombre entier) en choisissant publiquement un groupe fini G et un générateur g de ce groupe sans qu'un possible espion (Eve) ne puisse déterminer cette clé.

Le principe de ce protocole est assez simple :

- Alice et Bob choisissent un nombre premier p et un nombre entier a tel que $1 \leq a \leq p-1$.
- Alice choisit secrètement un nombre entier x_1 et Bob fait de même avec x_2.
- Alice calcule alors $y^1 = a^{(x_1)} \pmod{p}$ et Bob fait de même avec $y^2 = a^{(x_2)} \pmod{p}$.
- Alice et Bob s'échangent publiquement les valeurs de y_1 et y_2.
- Alice calcule $y_2^{x_1} = (a^{x_2})^{x_1} = a^{x_1 x_2} \pmod{p}$ et obtient K, la clé secrète.
- Bob calcule $y_1^{x_2} = (a^{x_1})^{x_2} = a^{x_1 x_2} \pmod{p}$ et obtient la même clé K que Alice.

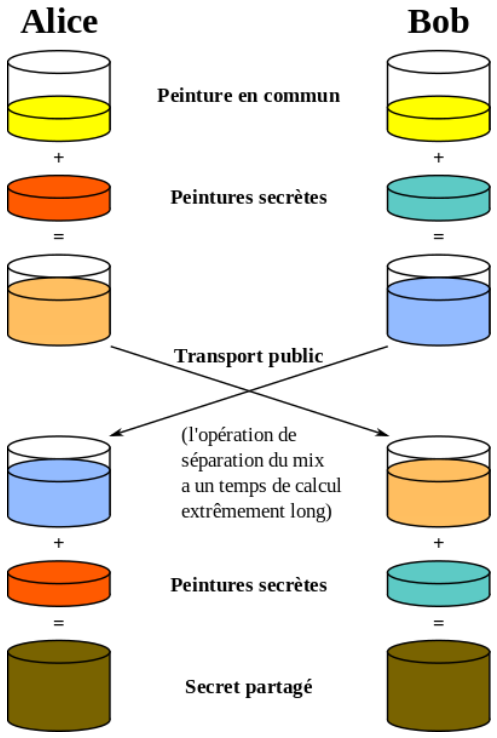

FIGURE 1 – Représentation Diffie-Hellman

D'après le problème du logarithme discret, Eve ne peut pas déterminer K en ayant connaissance de p, a, y_1 et y_2. Ainsi, Alice et Bob peuvent s'échanger en toute sécurité une clé de chiffrement K mise à part s'ils sont victimes d'une attaque dite de "l'homme du milieu" par Eve.

1.2 Chiffrement par transposition

Le chiffrement par transposition est un protocole d'échange de messages cryptés basée sur une méthode simple : permuter l'ordre des lettres du message suivant certaines règles.

Dans notre cas, nous allons ainsi créer une anagramme du message initial en respectant ces conditions en s'aidant d'un tableau :

- Pour commencer, nous allons créer un tableau de la largeur de la clé de chiffrement
- Nous écrirons ensuite caractère par caractère et ligne après ligne le message à coder
- Il suffit alors d'organiser les colonnes suivant l'ordre croissant des caractères de la clé de chiffrement
- Pour écrire le message chiffré, nous allons alors lire le tableau colonne par colonne en créant des mots de tailles similaires
- Afin de déchiffrer le message, il suffit de reprendre ces étapes à l'envers

FIGURE 2 – Exemple de transposition

Ce genre de chiffrement est très rudimentaire et peut facilement être craqué à l'aide d'études statistiques d'apparition de lettres dans le message codé pour définir la langue. Par la suite, on pourra déterminer la longueur de la clé et ainsi découper chaque mot pour essayer de trouver des anagrammes de chaque mot en utilisant un dictionnaire dans la langue déterminée auparavant.

2 Journal de bord

2.1 Séance 1

Lors de la première séance, notre choix s'est immédiatement tourné vers le sujet de cryptographie qui nous intéressait tout particulièrement. Après avoir pris connaissance du sujet, nous avons immédiatement entamé les recherches concernant l'échange de clé de Diffie-Hellman afin de comprendre son principe et commencer un début d'implémentation simple en java.

Nous avons pu poser quelques questions relatives au sujet afin de bien le comprendre et orienter nos travaux dans ce sens.

2.2 Séance 2

Durant cette séance, nous avons principalement continué nos recherches toujours axées sur l'échange de clé de Diffie-Hellman mais également sur le chiffrement par transposition. Un début de compte-rendu à commencé à être rédigé ainsi que le projet java final avec lequel les programmes seront implémentés a été créé.

2.3 Séance 3

Pendant cette séance, nous nous sommes penchés sur le problème de l'attaque de l'homme du milieu. Après quelques recherches, nous avons pu commencer son implémentation basique et ainsi répondre au prolongement n°3.

2.4 Séance 4

Lors de cette dernière séance, nous avons principalement recherché des informations à propos du problème du logarithme discret, son principe et son lien avec la méthode de Diffie-Hellman.

2.5 Hors séance

Si nous avons principalement fait des recherches sur les sujets de cryptographie durant les quatre séances de modélisation, la plupart du code à été fait en dehors des séances en s'appuyant sur nos travaux déjà réalisés..

Ce sujet a la particularité par rapport au sujet de la première phase de modélisation que nous avions choisi d'être beaucoup moins exigeants en terme d'implémentation. En effet, étant donné que les algorithmes de cryptographie existent déjà, le travail que nous avons s'est résumé à les comprendre et traduire ces algorithmes en java.

3 Implémentation

Lors de l'implémentation de l'algorithme de Diffi-Hellman, nous avons eu différents choix à faire, notamment une base g ainsi qu'un nombre premier p.

Dans notre cas, nous avons pris $g = 2$, en effet, même si celui-ci est un générateur à proprement parlé dans p, il permet de générer un nombre de valeurs bien assez grand pour assurer la pérennité de la clé de chiffrement.

Pour choisir un nombre premier p, étant donné la difficulté de recherche de très grands nombres premiers, nous avons donc dû utiliser la méthode probablePrime de la classe BigInteger de java. Celle-ci utilise un test de primalité probabiliste renvoyant un nombre premier ou proche d'un nombre premier. C'est-à-dire un nombre dont la décomposition en facteur premier serait faible ce qui n'est pas gênant dans notre cas puisque pour présenter une réelle faille de sécurité, il faudrait le découpage en facteur premier de p soit très grand.

```
1  import java.math.BigInteger;
2  import java.security.SecureRandom;
3
4  public class PerfectGenerator {
5      private BigInteger g;
6      private BigInteger p;
7
8      public PerfectGenerator() {
9          g = BigInteger.valueOf(2);
10         p = BigInteger.probablePrime(2048, new SecureRandom());
11     }
12 }
```

FIGURE 3 – Implémentation de PerfectGenerator

4 Modèle mathématique

4.1 Le problème du logarithme discret

Le principal modèle mathématique auquel nous avons été confrontés pour ce sujet est le problème du logarithme discret, base sur laquelle se construit l'échange de clé de Diffie-Hellman. Il s'avère que nous sommes actuellement incapables de le résoudre et c'est la raison pour laquelle il est largement utilisé en cryptographie à clé publique.

Après avoir étudié son fonctionnement, son principe repose sur le fait qu'il est impossible pour le moment de déterminer un entier l pour lequel $l = log_g y$ avec g étant un générateur d'une groupe cyclique G et $y \in G$. C'est essentiellement le fait que le groupe G choisi est un groupe cyclique et que $log_g y$ soit modulo n, n étant l'ordre du groupe G qu'aucune solution n'a été découverte pour résoudre ce problème.

4.2 Baby step giant step

Cet algorithme est une des solutions envisageables pour approcher la résolution du problème du logarithme discret. Avec celui-ci, d'autres algorithmes tels que le Rho de Pollard qui permet de décomposer en produit des nombres premier et l'algorithme de Pohlig-Hellman qui divise le problème du logarithme discret en autres problèmes de logarithme discret plus petits jusqu'à construire une solution. Le principe de l'algorithme baby step giant step est de résoudre le problème du logarithme discret en effectuant au maximum $2\sqrt{n}$ multiplications dans le groupe G.

On a p est un nombre entier premier.
On cherche alors le logarithme discret pour h.
On pose alors $x = logDiscret(h) = m * q + r$ avec q et r sont des nombres entiers.
On sait également que $0 \leq r < m$ et $m = \lceil \sqrt{n} \rceil$.

On obtient l'équation $g^x = g^{mq} g^r \mod p$ avec laquelle on peut caluler les pas de géants :
g^i pour i allant de 0 à $m - 1$ et on stocke la valeur du résultat dans un tableu associatif avec la clé i.

Une fois le tableau associatif des pas de géants remplis, nous pouvons initialiser q à 0 et calculer les pas de bébés suivant cette équation : $h(g^{-m})^q$ en prenant attention d'incrémenter q à chaque calcul. A chaque résultat obtenu, nous le comparons au tableau associatif des pas de géant. Dès qu'il y a égalité, on arrête les calculs et on obtients $logDiscret(h) = q * m + 1$ pour les dernières valeurs de q et m.

Bien que cette méthode permette de réduire à environ $2\sqrt{n}$ multiplications comparée à la recherche exhaustive de solution, on peut cependant préciser que $\sqrt{n}$ reste exponentiel.

Dans la pratique, l'utilisation de l'algorithme baby step giant step reste limité puisqu'il convient de trier le tableau des pas de géant et demande un espace mémoire permettant de charger un tableau de taille $\sqrt{n}$ environ. Autant dire que cela le rend inutilisable assez rapidement. C'est ici que nous pourrons proposer l'utilisation de l'algorithme de Rho de Pollard qui permet la résolution du problème du logarithme discret avec $\sqrt{n}$ multiplications et sans espace mémoire en utilisant le principe du paradoxe des anniversaires.

5 Conclusion

5.1 Les difficultés rencontrées

Tout au long de ce sujet, nous nous sommes rendu compte assez rapidement que dans le domaine de la cryptographie, la plus grande difficulté réside notamment dans le respect des différents modèles mathématiques lors de l'implémentation. En effet, tout manquement à quelques règles que ce soit peut engendrer des failles de sécurité et ne pas assurer la transmission de messages privés correctement.

Nous avons ainsi du "traduire" correctement chacun des algorithme utilisés afin de respecter les modèles mathématiques sur lesquels ils sont basés et grâce auxquels ils permettent la sécurité des données.

5.2 Les outils acquis

La crypto, c'est cho

Sommaire