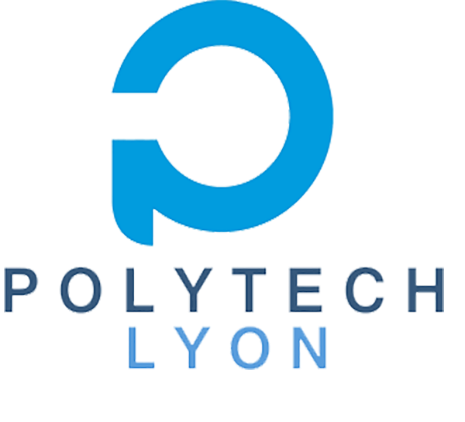
Projet Cryptographie

*Rapport*



# 1 - Présentation du projet

Le projet est constitué de 3 parties implémentant en Java des problèmes de cryptographie. Afin de résoudre ces problèmes, nous avons implémenté et utiliser le cryptosystème de Paillier car ces propriétés homomorphiques, con caractère probabiliste et sa sécurité sémantique sont très utiles.

La classe Paillier est constituée des méthodes suivantes :

- keyGen() : renvoie pk et sk

- encrypt(pk, m) : renvoie em une encryption du message m selon la clef publique pk

- encryptPlus(pk, m) : renvoie M et r, avec r un entier aléatoire utilisé pour l’encyption

- decrypt(pk, sk, em) : renvoie m le résultat du décryptage de em

- decryptPlus(pk, sk, em) : renvoie m et r

# 2 - Protocole Multiplication

## Contexte

Le protocole de multiplication implémenté permet à Alice d’obtenir l’encryption du produit de deux messages x et y dont elle a uniquement les encryptions X=Encrypt(pkB, x), Y=Encrypt(pkB, y).

Pour ce faire, elle communique sur le réseau avec Bob dont elle connait la clef publique pkB.

## Implémentation

La classe Multiproof a pour attribut une instance de la classe Paillier, une méthode multiproof(XA, YA, pkB, skB). Cette dernière déroule le protocole instauré entre Alice et Bob en 2 phases et renvoie XYA = enccypt(pk, xA\*yA) si le protocole est validé, 0 sinon.

Phase 1 : Calcul de l’encryption du produit

Phase 2 : Preuve du calcul réalisé par Bob pour Alice.

Une nomenclature des variable permet de connaitre à qui elles appartiennent ; elle sont nommées du suffixe A pour Alice et B pour Bob. De plus, les variables encryptés commencent par une majuscule.

**Phase 1**

Alice génère sA et tA aléatoirement afin de masquer X et Y qui doivent rester inconnus de Bob, puis envoie UA=XA \* SA et VA = YA \* TA.

Grace à l’homomorphisme additif de Paillier, Bob peut les décrypter et obtenir uB = xA+sA et vB = yB+tB. Il renvoie alors à Alice l’encryption WB = encrypt(pkB, uB\*vB)

Alice peut alors en déduire facile XYA ( Encrypt(pk, xA\*yA)) en utilisant les propriétés homorphiques.

**Phase 2**

Le but de cette phase est que Bob prouve qu’il a bien réalisé un produit et qu’il l’a envoyer à Alice.

Pour ce faire, ils échange certaines valeurs encryptés et d’autres générés aléatoirement par Alice. Bob va devoir effectuer devoir décrypter des message comprenant les termes UA et VA. Il utilisera la methode DecryptPlus(…) afin d’envoyer à Alice les messages décrypté et les r.

Pour finir, Alive pourra tester 3 assertions. Si l’une d’entre elles est fausse, cela signifie que Bob à trahi le protocole.

La classe Multiproof contient également une méthode main() à exécute pour lancer le protocole.

Dans cette méthode, on définit aléatoirement x et y puis les clefs pk, sk de Bob pour finalement encrypter x et y.

Ensuite on fait appel à la méthode multiproof(…).

Une vérification du produit est réalisée à la fin pour confirmer l’algorithme.

# 3 - Calcul 3 DNF

TODO

# 4 - Bracelet electronique

TODO