Systèmes et réseaux optimisés

Rapport de projet

Implémentation d'une version parallèle du mode CTR

I. Objectifs

Le but de ce projet est d'apprendre à utiliser les threads et de mettre en évidence l'importance de l'utilisation de ces tâches parallèles en terme de temps d'exécution.

II. Implémentation

Pour réaliser ce type de chiffrement, nous avons utilisé l'algorithme de chiffrement par bloc 'Rijndael', une clé secrète de 128 bits et un compteur sur 128 bits. Le fichier à chiffrer est découpé en blocs de 128 bits.

Notre programme comporte 3 classes :

– Main.java qui contient :

les blocs de 128 bits à chiffrer :

```
// Découpage du texte en blocs de 128 bits :
File plainText = new File("test.txt");
byte[] blocInf128 = new byte[new FileInputStream(plainText).available() % 16];
byte[][] blocs128 = Utils.splitFile(plainText, blocInf128);
cipheredBloc128 = new byte[blocs128.length][blocs128[0].length];
System.out.println("Fichier decoupé en "+blocs128.length+" blocs de 128 bits + 1 bloc de "+ blocInf128.length+" bytes");
System.out.println("Soit une taille de " + (blocs128.length * 16 + blocInf128.length) + " octets");
```

un vecteur d'initialisation qui joue le rôle de compteur :

```
// Contiendra tous les IV incrémentés :
byte[][] initializationVectorIncremented;
if(blocInf128.length == 0){ initializationVectorIncremented = Utils.ivsIncremented(blocs128.length, 16); }
else{initializationVectorIncremented = Utils.ivsIncremented(blocs128.length + 1, 16);}
```

une clé secrète :

```
// Génération de la clé :
byte[] secretKeyInByte = new byte[16];
new SecureRandom().nextBytes(secretKeyInByte);
```

l'initialisation de l'algorithme de chiffrement par bloc 'Rijndael' :

```
// Algorithme de chiffrement par bloc :
Rijndael rijndael = new Rijndael();

Object secretKey = null;
try {
    secretKey = rijndael.makeKey(secretKeyInByte, 32);
} catch (InvalidKeyException e1) { e1.printStackTrace(); }
```

la gestion des threads :

```
for(byte[] unBloc : blocs128){
    /*
     * 'numberOfThread' threads peuvent être executés en meme temps, il faut attendre la fin des autres sinon.
     */
    /**
     * TODO: Cette partie de la gestion des threads est mal optimisée.
     */
    while(!threadIsFree){
        for(indiceThreadNotAlive = 0 ; indiceThreadNotAlive < tabThread.length && !threadIsFree ; indiceThreadNotAlive++){

            if(tabThread[indiceThreadNotAlive].getState() != Thread.State.RUNNABLE && tabThread[indiceThreadNotAlive].getState() != Thread.State.TIMED_WAITING ){
                threadIsFree = true;
            }
        }
    }
    tabThread[indiceThreadNotAlive -1] = new Thread(new ThreadCTR( rijndael,
                                                                   unBloc,
                                                                   secretKey,
                                                                   initializationVectorIncremented[numBlocEncours] ));
    tabThread[indiceThreadNotAlive - 1].start();

    threadIsFree = false;
}

// On attend que tous les threads se terminent
for(int i=0 ; i<tabThread.length ; i++){
    try {
        tabThread[i].join();
    } catch (InterruptedException e) {e.printStackTrace();}
}
```

Un nombre déterminé de threads peuvent êtres en cours d'exécution. Ces threads sont stockés dans un tableau. Si un thread a terminé son exécution, alors un nouveau thread est immédiatement créé.

- ThreadCTR qui définit le chiffrement en mode CTR en utilisant l'algorithme de 'Rijndael'.
  Cet algorithme est définit comme suit :
  - chiffrement du compteur avec la clé secrète en utilsant 'Rijndael'
  - XOR du résultat avec le bloc de texte clair.

```java
public class ThreadCTR implements Runnable{

    private byte[] plainTextBloc;
    private byte[] initializationVectorIncremented;
    private byte[] cipheredBlock = new byte[16];
    private Object secretKey;
    private Rijndael rijndael;

    public ThreadCTR(){}

    public ThreadCTR(Rijndael rijndael, byte[] plainTextBloc, Object secretKey, byte[] initializationVectorIncremented) {
        this.plainTextBloc = plainTextBloc;
        this.secretKey = secretKey;
        this.initializationVectorIncremented = initializationVectorIncremented;
        this.rijndael = rijndael;
    }

    @Override
    public void run() {
        assert(rijndael != null);
        assert(plainTextBloc != null);
        assert(secretKey != null);
        assert(cipheredBlock != null);

        // Chiffrement par block :
        rijndael.encrypt(initializationVectorIncremented, 0, cipheredBlock, 0, secretKey, 16);
        assert(cipheredBlock.length > 0);
        assert(cipheredBlock.length == plainTextBloc.length);

        // XOR :
        for(int i = 0 ; i < plainTextBloc.length ; i++){
            cipheredBlock[i] = (byte) (cipheredBlock[i] ^ plainTextBloc[i]);
        }
    }
}
```

III. Résultats / Améliorations

Notre partie sur la gestion des threads n'est pas efficace. La JVM passe plus de temps à vérifier qu'un thread est toujours en exécution plutôt que de chiffrer les données.

Utilisation : lancer le programme projetCTR.jar dans la console. Vérifier que le fichier « test.txt » est bien présent au même niveau que le programme.