

Encadrant :

POPINEAU Fabrice

BOUCHER Julien

MAISON Jonas

|  |
| --- |
| CentraleSupélec Campus de Gif, Année 2016-2017  Rapport de projet de développement logiciel :  Cryptographie et arbres de Merkle |

**GitHub :** <https://github.com/Jonas1312/merkle-tree>

Sommaire

[Sommaire 2](#_Toc481680296)

[Introduction 3](#_Toc481680297)

[Signature de Lamport 4](#_Toc481680298)

[1. Génération de la clé privée 4](#_Toc481680299)

[2. Génération de la clé publique 4](#_Toc481680300)

[3. Génération de la signature du message 4](#_Toc481680301)

[4. Vérification de la signature 5](#_Toc481680302)

[5. Avantages et inconvénients 5](#_Toc481680303)

[Arbre de Merkle 6](#_Toc481680304)

[Conception du logiciel 7](#_Toc481680305)

[1. Génération des classes utilisées 7](#_Toc481680306)

[Conclusion 8](#_Toc481680307)

[Références 9](#_Toc481680308)

Introduction

La signature numérique est un terme générique qui indique tout simplement un mécanisme permettant, comme une signature papier, de démontrer que le fichier ou document signé est bien original et n’a pas été altéré ou modifié par un tiers. Elle permet entre autre de vérifier l'identité de l'auteur des informations numériques, tels que des documents, des messages électroniques ou des transactions monétaires à l'aide d'algorithmes de chiffrement.

Une signature numérique doit respecter les conditions suivantes :

* Infalsifiable : elle ne peut pas être falsifiée par un tiers.
* Authentique : elle permet de garantir que le signataire est bien celui qu’il prétend être.
* Inaltérable : une fois que le document est signé on ne peut plus le modifier.
* Irrévocable : la personne qui a signé ne peut le nier.
* Non réutilisable : elle ne doit être utilisée qu’une fois pour authentifier un document.

Contrairement aux idées reçues, la signature numérique ne chiffre ou ne crypte pas le document, il reste visible et lisible à n’importe quel utilisateur présent sur le réseau de communication.

Dans le cadre de notre projet de développement logiciel nous avons étudié un schéma de signature basé sur l’arbre de Merkle (ou arbre de hashage), qui est une structure de données utilisée notamment pour différents protocoles ou logiciels tels que Git, Bittorrent ou le Bitcoin, qui reposent tous sur l’utilisation d’un système de signature à usage unique pour authentifier les échanges de données.

Les arbres de Merkle ont été développés dans les années 70 par Ralph Merkle, et permettent principalement d’économiser le nombre de clés publiques nécessaires pour transmettre des messages. Un autre avantage est qu’il semblerait que le système de signature de Merkle soit résistant aux attaques d’algorithmes quantiques.

Dans un premier temps nous expliquerons dans ce rapport le fonctionnement de la signature de Lamport, qui est souvent utilisée avec les arbres de Merkle que nous étudierons ensuite. Puis nous verrons le démonstrateur qui a été conçu, permettant de mettre en œuvre la signature et la transmission de données sur le réseau, et enfin nous conclurons.

Signature de Lamport

Le schéma de signature de Lamport, inventé en 1979 par Leslie Lamport, est une méthode permettant de créer des signatures numériques. La sécurité des signatures de Lamport repose sur des fonctions de hachages cryptographiques, aussi appelées « one way function ».

Une fonction de hachage est une fonction qui est facile à calculer mais qui est très difficile à inverser. Ces fonctions sont en général utilisées pour générer des empreintes uniques de fichier, mais sont aussi fortement utilisées dans le domaine de la cryptographie.

Le schéma de signature de Lamport se déroule en quatre étapes : génération d’une clef privée, génération d’une clef publique, génération d’une signature pour un message donné, et enfin vérification de la signature du message par le destinataire du message.

On suppose qu’Alice souhaite envoyer un message « M » ainsi qu’une signature « sig » à Bob. Elle dispose d’un générateur de nombre aléatoires et d’une fonction de hash cryptographique sur 256 bits telle que le SHA256.

### Génération de la clé privée

Pour créer une clef publique d’une signature de Lamport, Alice utilise un générateur de nombre aléatoires pour générer 256 paires de nombres aléatoires, chaque nombre étant défini sur 256 bits. C’est la clef privée d’Alice qui doit absolument rester privée. La taille de cette clef est de 2x256x256 bits, c’est-à-dire 16 KiB au total.

|  |  |  |
| --- | --- | --- |
|  | X0 | X1 |
| 0 | 256 bits | 256 bits |
| 1 | 256 bits | 256 bits |
| … | 256 bits | 256 bits |
| 255 | 256 bits | 256 bits |

### Génération de la clé publique

Pour créer la clef publique, il suffit de calculer les hashs de chaque nombre de la clef privée (512 nombres donc). La fonction de hachage générant des nombres de 256 bits, la taille de la clef publique est identique à celle de la clef privée, soit 16 KiB.

### Génération de la signature du message

Dans un premier temps Alice utilise la fonction de hachage pour générer le hash du message à signer. Ensuite, pour chaque bit du hash précédemment calculé Alice prend le nombre de la colonne X0 de la clef privée si le bit est 0, sinon elle prend le nombre de la colonne X1 de la clef privée. Cela produit une séquence de 256 nombres de 256 bits, soit 8 KiB, c’est la signature du message.

Maintenant que la clef privée d’Alice a été utilisée, elle peut être détruite car elle ne doit pas être transmise et elle ne doit plus jamais servir sous peine de rendre vulnérable la sécurisation des messages.

Alice transmet à Bob le message, la signature de Lamport et la clef publique utilisée pour générer la signature.

### Vérification de la signature

Bob souhaite vérifier que le message est bien authentique et n’a pas été modifié lors du transfert. Pour cela, il commence par générer un hash du message et calcule les hashs de chacun des 256 nombres de la signature de Lamport. Pour chaque bit du hash du message, si la valeur du bit est 0 on prend le nombre de la colonne X0 (ou X1 si le bit est à 1) de la clef publique et on le compare au hash correspondant de la signature de Lamport, si les deux sont égaux on répète l’opération 256 fois. Si tous les hashs correspondent le message est authentique, sinon il y a eu une modification du message de la signature ou de la clef publique.

### Avantages et inconvénients

La sécurité des signatures de Lamport dépend directement de la fonction de hachage utilisée ainsi que du générateur de nombre aléatoire utilisé pour générer la clef privée. En effet, certaines fonctions de hachage comme le MD5 ne sont pas parfaitement fiables et présentent des collisions (deux fichiers ayant le même hash). La signature de Lamport présente cependant l’avantage d’être facilement adaptable à n’importe quelle fonction de hachage.

L’inconvénient majeur des signatures de Lamport est qu’un couple de clefs publiques et privées ne peut être utilisé qu’une seule fois pour signer un message. De plus, la taille des données à transmettre est assez importante quel que soit la taille du message (16 KiB pour la clef publique et 8KiB pour la signature). Pour pallier à ce problème, on utilise les arbres de Merkle qui permettent d’utiliser une seule clef publique pour signer plusieurs messages.

Arbre de Merkle

Comparaison tailles signature

Utilisation bitcoin, git

Conception du logiciel

### Génération des classes utilisées

Pour implémenter l’algorithme de signature basé sur les arbres de Merkle, nous avons dû créer les classes MerkleTree, LamportSignature, ainsi que les classes Client et Server pour la partie simulation de réseau.

* MerkleTree :

Cette classe possède les attributs *tree*, *n\_levels* et *n\_leaves*. Le premier représente l’arbre sous forme de dictionnaire de tuples (*position*, *data*), position étant lui-même un tuple (*level*, *index*). On compte dans l’arbre de bas en haut et de gauche à droite, la racine est donc à la position (*level\_max*, *0*). *n\_levels* représente la hauteur de l’arbre ( on compte à partir de 0, une racine simple est donc de hauteur 0). Enfin, *n\_leaves* représente le nombre de feuilles de l’arbre.

Quant aux méthodes de MerkleTree :

* **add\_node**(*data, position, hashed*) permet d’ajouter un nœud avec la valeur *data* en position *position* de l’arbre.
* **generate\_tree**() permet de générer l’arbre entier à partir des feuilles
* **get\_root**() nous donne la racine de l’arbre
* **get\_brother\_node\_hash**() renvoie le hash du nœud voisin (le deuxième fils du nœud père)
* **get\_brother\_node\_position**() renvoie la position du nœud voisin
* **get\_authentification\_path**(*index*) renvoie le chemin d’authentification en partant de la feuille *index*
* **get\_authentification\_path\_hashes**(*index*) renvoie les hashs du chemin d’authentification en partant de la feuille *index*
* **hash**(*data*) renvoie le hash de *data*

Conclusion

Références

[1] RALPH C. MERKLE. A Digital Signature Based On A Conventional Encryption Function

[2] LESLIE LAMPORT. Constructing Digital Signatures from a One Way Function

[3] Georg Becker. Merkle Signature Schemes, Merkle Trees and Their Cryptanalysis