Université Abou Bekr Belkaid
Tlemcen Algérie
جامعة أبي بكر بلقايد
تلمسان الجزائر

République Algérienne Démocratique et Populaire
Université Abou Bakr Belkaid– Tlemcen
Faculté des Sciences
Département d'Informatique

Mémoire de fin d'études

Pour l'obtention du diplôme de Master en Informatique

Option: *Modèle intelligent et décision (M.I.D)*

Thème

Optimisation des HMAC par les réseaux de neurones

Réalisépar :

- **Mr MEGHELLI Mohammed Yassine**

Présenté le 09 Octobre 2013 devant le jury composé de MM.

- *MrBenmaamarBadr.* *(Président)*
- *MrZiani-CherifSalim.* *(Encadreur)*
- *Mme LABRAOUI Nabila.* *(Examinateur)*
- *MrHadjilaFethallah.* *(Examinateur)*

Annéeuniversitaire:2012-2013

Table des matières

INTRODUCTION GÉNÉRALE : 1

CHAPITRE I : HACHAGE ET AUTHENTIFICATION DES MESSAGES 3

1. INTRODUCTION : 4
2. LES FONCTIONS DE HACHAGE : 4
2.1. FONCTIONNEMENT 4
2.2. PROPRIÉTÉS : 5
2.3. OBJECTIFS CRYPTOGRAPHIQUES : 5
2.4. SECURE HASH ALGORITHM: 6
2.4.1. L'algorithme SHA-256 : 6
2.4.2. Schémas fonctionnel interne du SHA-2 : 7
2.5. LE NN-HASH 7
2.5.1. Généralités sur les réseaux de neurones : 8
2.5.2. La fonction de hachage : 9
2.5.3. Réseau de neurones utilisé : 10
2.5.4. Fonction d'activation des neurones : 11
2.5.5. Modes de hachage : 13
2.5.6. Sécurité : 13
3. MESSAGE AUTHENTICATION CODE: 14
3.1. HASH-BASED MESSAGE AUTHENTICATION CODE: 15
4. ATTAQUES CRYPTANALYTIQUES: 16
4.1. LE BRUTE-FORCE : 16
4.2. ATTAQUE PAR PARADOXE DES ANNIVERSAIRES : 16
4.3. ATTAQUES PAR PRÉ-IMAGE : 17
4.4. RÉSISTANCE DES FONCTIONS DE HACHAGE AUX ATTAQUES CRYPTOGRAPHIQUES 17
5. CONCLUSION : 18

CHAPITRE II : VERS UN HMAC PLUS OPTIMAL ERREUR ! SIGNET NON DEFINI.

1. INTRODUCTION : 20
2. SCHÉMAS GÉNÉRAL : 20
3. ADAPTER LE NN-HASH POUR LE RÉ-HACHAGE DES SIGNATURES SHA256 : 21
3.1. FONCTION GÉNÉRALE DE TRANSFERT : 22
3.2. FONCTIONS DE TRANSFERT DES DIFFÉRENTES COUCHES : 23
3.3. FONCTION DE HACHAGE : 24
4. ANALYSE DES PERFORMANCES : 24
4.1. GAINS DE L'OPTIMISATION DU HMAC : 24
4.2. SENSIBILITÉ DU NN-HMAC : 25
5. ANALYSE DE SÉCURITÉ : 26
5.1. RÉSISTANCE À L'ATTAQUE PAR BRUTE-FORCE : 26

5.2. RÉSISTANCE À L'ATTAQUE PAR PARADOXE DES ANNIVERSAIRES : 26
5.3. RÉSISTANCE À L'ATTAQUE PAR PRÉ-IMAGE : 26
6. DISCUSSION ET CRITIQUES : 27
7. CONCLUSION : 28

CONCLUSION GÉNÉRALE ET PERSPECTIVES : 29

BIBLIOGRAPHIE 30

Liste des Tables et Figures :

Figure 1-1: Fonction de hachage. 4
Figure 1-2. : Tour de hachage des Algorithmes SHA-2 7
Figure 1-3. : Réseau de neurones multicouche 8
Figure 1-4. : Neurone formel 9
Figure 1-5. : Fonction de hachage du NN-Hash 9
Figure 1-6. : Réseau de neurones du NN-Hash 11
Figure 1-7. : Utilisation des MAC 14
Tableau 1-1. : Quelques mécanismes MAC 15
Figure 1-8. : Schéma général du HMAC 15
Tableau 1.2. : Quelques algorithmes de hachage et leurs résistances. 17
Figure 2-1. : Schéma général du modèle proposé. 20
Figure 2-2. : Réseau de neurone du NN-Hash adapté. 22
Tableau 2-1. : Comparaison du HMAC-SHA-256 et du NN-HMAC-SHA-256. 24
Tableau 2-2. : Gains d'optimisation du NN-HMAC-SHA256 25
Figure 2-3. : Sensibilité du modèle proposé 25

Introduction générale :

En cryptographie, le hachage est la fonction par laquelle on produit une petite imitation représentative et unique d'une donnée quelconque. Le haché, produit de cette fonction, est utilisé pour des tâches d'identifications rapides et/ou sécurisées.

La propriété la plus importante des fonctions de hashage étant le chiffrement à sens-unique, et grâce à l'implicite de cette propriété dans les réseaux de neurones, un algorithme de hachage peut être construit. En effet, des chercheurs de l'université de Nanjing ont publié en 2007 un article spécifiant une fonction de hachage basée sur les réseaux de neurones baptisée NN-Hash [5].

Les différentes contraintes et les divers emploies autour de l'identification ont permis de donner naissance à de nouveaux moyens autres que (ou basés sur) le hachage. MAC, ou *Message Authentication Code*, est l'un de ces nouveaux moyens, son code vérifie l'authenticité et l'intégrité des données pour deux ou plusieurs entités communicantes, il aide à satisfaire des objectifs tel que l'anonymat, la confidentialité, l'authentification... etc.

Selon les besoins, des mécanismes et des algorithmes ont été créés pour calculer les MAC. Le Checksum, par exemple, est une simple fonction de sommation par laquelle on peut détecter une altération accidentelle lors d'un échange de données. D'autres mécanismes comme le HMAC, peuvent grâce aux fonctions de hachage, détecter aussi des altérations d'ordres intentionnels.

HMAC est un mécanisme dont les paramètres sont une clé secrète partagé ainsi qu'un algorithme sous-jacent de hachage [9]. Il génère, par une double computation, un code du même niveau de sécurité et de la même taille que celles du haché de la fonction utilisée. Considéré comme une 'pseudo' fonction de hachage, il est préférable qu'une implémentation du HMAC soit (dans la mesure du possible) rapide et optimale.

Nous proposons par notre travail, un modèle qui pourrait optimiser les HMAC. Pour ce faire, on modifie le NN-Hash afinqu'il prenne la place de la deuxième computation, et cela, pour fixer la taille du haché (quel que soit la fonction utilisée), gagner en complexité de calcul et avoir une sécurité acceptable.

Notre travail est organisé en deux chapitres, le premier présente les notions de bases entourant le hachage et l'authentification, le second spécifiera notre modèle par une application, utilisant SHA-256 comme sa fonction sous-jacente de hachage, ainsi qu'une modeste analyse sur les résultats obtenus.

Chapitre I

Hachage et authentification des messages

1. Introduction :

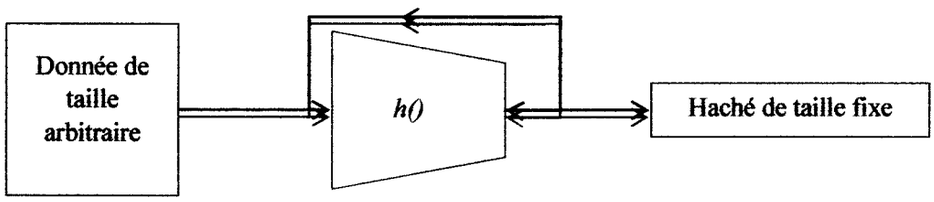
Le hachage et l'authentification des messages sont des outils cryptographiques motivés par l'identification rapide et/ou sécurisée d'entités. La grande variété de contraintes sur l'identification a encouragé la création de plusieurs crypto-systèmes par la pratique d'une multitude d'approches.

Ce chapitre nous aide à comprendre fonctionnement, contraintes et objectifs du hachage et de l'authentification de message tout en exposant des caractériels d'algorithmes et de mécanismes susceptibles de participer à l'optimisation d'un standard de l'authentification des messages.

2. Les fonctions de hachage :

Une fonction de hachage est un outil de cryptographie capable de fournir pour une donnée quelconque d'une taille quelconque, une empreinte unique de taille fixe.

Par un résultat de confusion/compression des données initiales, une fonction de hachage se doit de garantir la confidentialité, l'optimisation ou l'intégrité de ces données, et cela, dans un temps minime et avec une sécurité optimale (selon les besoins).



h() : Fonction Itérative de confusion et de compression

Figure 1-1: Fonction de hachage.

2.1. Fonctionnement

Pour compresser et crypter les données, une fonction de hachage se base, selon les besoins, sur le chiffrement en bloc ou sur le chiffrement de flux. Le chiffrement utilise

ensuite une table de constantes de hachage ainsi qu'une séquence d'opérations simples tel que l'addition, la multiplication ou encore les opérations sur les bits de données.

L'algorithme de hachage doit être conçu de façon à ce qu'un changement minime des données initiales entrainerait une perturbation conséquente sur l'empreinte générée.

2.2. Propriétés :

Les fonctions de hachage possèdent plusieurs propriétés, les plus importantes sont :

Chiffrement à sens unique (résistance à l'attaque de pré-image) :

Sachant que le hachage est une application surjective, il ne doit pas y avoir une fonction inverse qui permettrait la restauration des données à partir de l'empreinte.

Résistance à l'attaque de seconde pré-image :

En présence de la fonction de hachage et d'un couple message/empreinte généré par cette dernière, un attaquant ne peut générer la même empreinte avec un autre message.

Insensibilité aux données:

Une fonction de hachage doit traiter des données de n'importe quelle nature et de n'importe quelle taille (théoriquement infinies).

Résistance aux collisions :

Une fonction de hachage est censée théoriquement n'avoir aucune empreinte qui peut être généré par deux messages distincts.

2.3. Objectifs cryptographiques :

Les fonctions de hachage peuvent satisfait plusieurs buts cryptographiques dont :

L'Authenticité :

L'Authenticité est une forme d'identification, elle est utilisée pour la confirmation d'identité d'entités de différentes natures.

La signature numérique :

La signature numérique est un mécanisme qui permet de garantir l'intégrité d'une donnée en authentifiant son auteur.

L'Anonymat :

Les algorithmes de hachage peuvent offrir un service d'identification, d'authentification et de communication pour des personnes ne désirant pas donner certaines informations lors d'un échange quelconque.

D'autres objectifs ont eux aussi étés satisfaits par le hachage, comme le transfert oublieux[1], l'horodatage[2], la traçabilité[3]... etc.

2.4. Secure Hash Algorithm:

SHA est une famille de fonctions de hachage publiée pour la première fois en 1993 sous le standard FIPS-180 par l'agence du NIST[4]. Les trois premiers algorithmes (SHA-0, SHA-1, SHA-2) ont été conçus par la NSA[5]; le dernier de cette famille, le SHA-3, a été conçu suite à un concours public lancé par la NSA pour remédier à de probables faiblesses sur SHA-2.

2.4.1. L'algorithme SHA-256 :

SHA-256 appartient à la sous-famille d'algorithmes de hachage SHA-2, elle fut spécifiée en 2002 sur la publication FIPS-180-2, et demeure avec SHA-512 comme standard du gouvernement fédéral des États-Unis. Les algorithmes d'SHA-2 sont basés sur le fonctionnement de ces prédécesseurs SHA-0 et SHA-1 et inspiré par l'algorithme MD4[6].

SHA-256 est une fonction itérative dont les calculs sont modulo 2^{32}, elle est résistante à toutes les attaques réussie sur SHA-1, et ça, malgré la ressemblance qui existe entre les deux algorithmes en terme de fonctionnement.

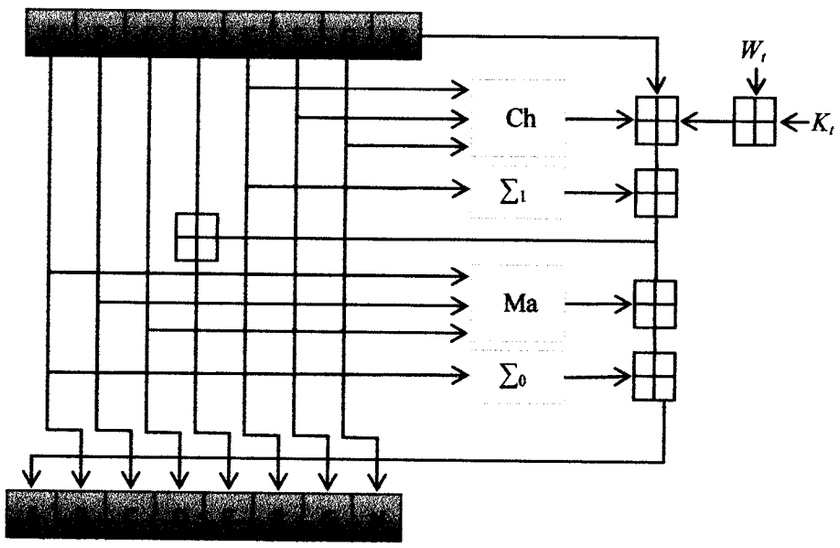
Caractéristiques [1] :

- Taille du message :2^{64} bits maximum

[1] Transfert oublieux : Communication par informations confidentielles implicite.
[2] Horodatage : Chiffrement des données en associant l'heure et la date d'un évènement.
[3] Traçabilité : Disposition d'information permettant de connaitre l'origine d'un produit ou d'un matériau.
[4] NIST : National Institute of Standards and Technology (Etats-Unis).
[5] NSA : National Security Agency (Etats-Unis).
[6] MD4 : Algorithme de hachage 'Message Digest 4'.

- Taille des blocs : 512 bits
- Taille des mots : 32 bits
- Taille du condensé : 256 bits
- Niveau de sécurité attendu (collision[7]) : 2^{128} bits
- Nombre de tours (fonction de compression) : 64

2.4.2. Schémas fonctionnel interne du SHA-2 :



A, B, C, D, E, F, G et H sont des mots de 32 bits (SHA-256) ou 64 bits (SHA-512) ;
$\boxplus$ est l'addition modulo 2^{32} (SHA-256) ou 2^{64} (SHA-512) ;
$Ch(E,F,G) = (E \wedge F) \oplus (\neg E \wedge G)$
$Ma(E,F,G) = (A \wedge B) \oplus (A \wedge C) \oplus (B \wedge C)$
$\sum_0(A) = (A \ggg 2) \oplus (A \ggg 13) \oplus (A \ggg 22)$
$\sum_1(E) = (E \ggg 6) \oplus (E \ggg 11) \oplus (E \ggg 25)$
Kt est une constante (32 ou 64 bits) qui dépend du numéro de tour t ;
Wt est un mot de 32 (SHA-256) ou 64 bits (SHA-512) qui dépend du numéro de tour t ; il est obtenu par une procédure d'expansion à partir du bloc de donnée (512 ou 1024 bits) dont le traitement est en cours.

Figure 1-2. : Tour de hachage des Algorithmes SHA-2

2.5. Le NN-Hash

Publié en 2007 dans l'article "One-way Hash FunctionBased on Neural Network" [5], Le NN-Hash est une fonction de hachage à clé qui a profité des propriétés des réseaux

[7]Collision : situation dans laquelle deux données ont un résultat identique avec la même fonction de hachage.

de neurones pour appliquer une diffusion/confusion/compression des données en entrée. Ce Modèle présente une résistance à plusieurs attaques cryptographiques, une sensibilité complète aux changements des données d'entrée et une haute sensibilité au changement de la clé utilisateur. Cette fonction génère des hachés de 128bits à partir de blocs de 1024.

2.5.1. Généralités sur les réseaux de neurones :

En intelligence artificiel, le modèle de calcul inspiré du fonctionnement biologique des neurones est appelé réseau de neurones artificiel. Il est utilisé dans divers domaines comme l'apprentissage artificiel ou comme pour le NN-Hash, la compression et la confusion de données…

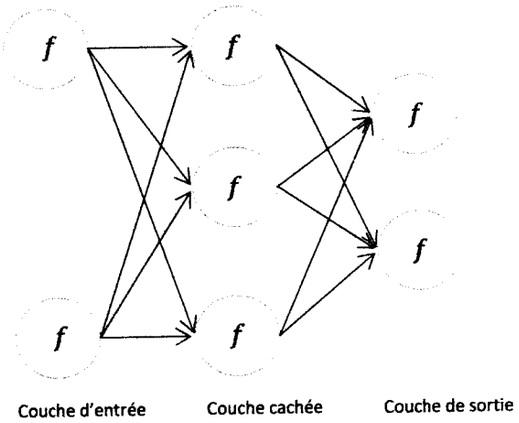
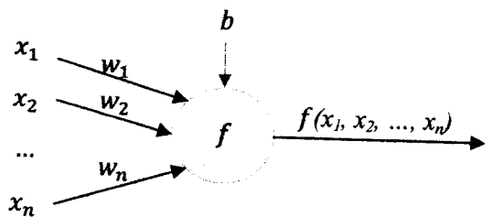


Figure 1-3. : Réseau de neurones multicouche

Un réseau de neurone dit multicouche, est composé d'une couche de neurones d'entrée qui diffuse les données vers la couche suivante, une ou plusieurs couches cachés qui détecte les caractéristiques des données diffusé par la couche précédente et une couche de sortie qui calcule le ou les résultats finaux. Chaque couche possède un nombre préétablie de neurones par rapport aux performances attendues. Les liens entre les neurones des différents couches sont appelés liens synaptiques, chaque lien possède un poids qui lui est propre.

Le neurone formel est l'unité fonctionnelle d'un réseau de neurone. Il est une



représentation formelle ...

Figure 1-4. : Neurone formel

La fonction d'activation des neurones (*f* sur la *figure I.3*) est la fonction par laquelle un neurone calcul sa sortie en utilisant ses entrées, ses différents poids synaptiques w_i (pour la pondération du calcul) et un biais b (pour le contrôle des résultats).

Plusieurs fonctions d'activation peuvent être affectées aux neurones, la plus simple est la somme pondérée des entrées du neurone.

2.5.2. La fonction de hachage :

En plus du réseau de neurones, la fonction de hachage du NN-Hash comporte un générateur de clés. En utilisant la clé-utilisateur, le générateur de clé produit les paramètres de contrôle (Q_i), les biais (B_i) ainsi que les poids (W_i) pour l'exécution du réseau de neurone proposé pour ce modèle-là. Pour y arriver, celui-ci subdivise la clé-utilisateur en 4 data-pixels K_0, K_1, K_2 et K_3pour les incorporer au calcul.

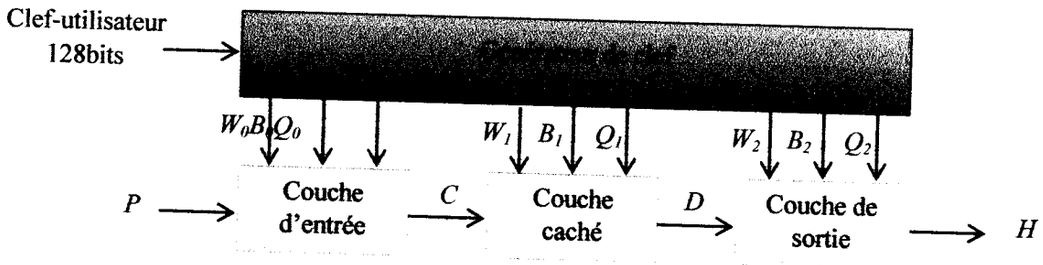


Figure 1-5. : Fonction de hachage du NN-Hash

Le générateur de clef est exprimé mathématiquement comme suit :

$$\begin{cases} X_0(k) = f^{T+k}(K_0, K_1) \\ X_1(k) = f^{T+k}(K_2, K_3) \\ K_s(k) = \big(X_0(k) + X_1(k)\big) \, mod \, 1 \end{cases}$$

$K_s(k)$ est un réel qui représente la k-ème sous clé (poids W_i, Biais B_i et paramètres de contrôle Q_i), ce modèle en comporte 151.

2.5.3. Réseau de neurones utilisé :

Le chiffrement à sens unique recherché par ce modèle élimine la nécessité de la rétro-propagation pour le réseau de neurones. Les neurones de ce réseau sont donc activé qu'une seule fois. Le réseau de neurone utilisé est composé de trois couches :

Une couche d'entrée composée de huit neurones :

La matrice P des données divisées en 32 data-pixels (quantifiés modulo 2^{32}) par bloc de 1024bits (à la fois) active les neurones de cette couches et renvoie la matrice C pour la seconde.

Une couche cachée composée de huit neurones :

La matrice C en sortie de la couche d'entrée active les neurones de cette couche et renvoie la matrice D pour la dernière couche.

Une couche de sortie composée de quatre neurones :

La matrice D en sortie de la couche cachée active les neurones de cette couche et renvoie 4 data-pixels du haché final en sortie.

Le haché final H est exprimé par :

$$H = f_2(W_2D + B_2) = f_2(W_2f_1(W_1C + B_1) + B_2) = f_2(W_2f_1(W_1f_0(W_0P + B_0) + B_1) + B_2)$$

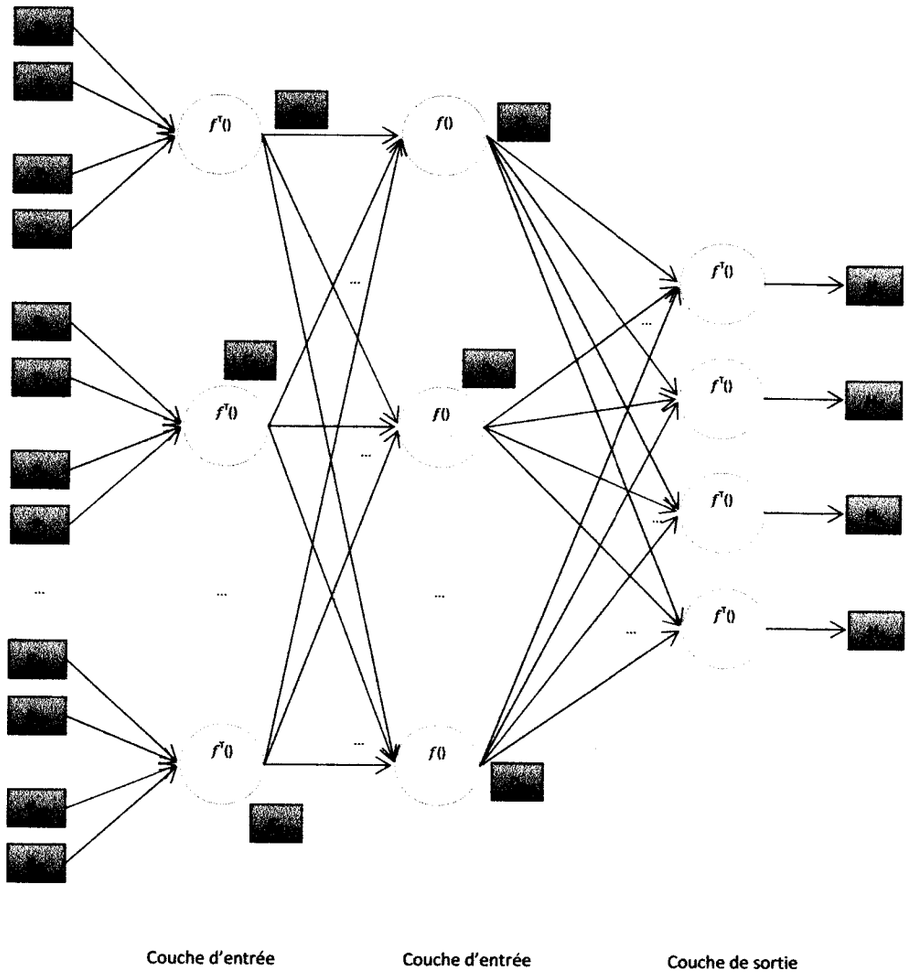


Figure 1-6. : Réseau de neurones du NN-Hash

2.5.4. Fonction d'activation des neurones :

$$f(X(k),Q)=\begin{cases}\dfrac{X(k)}{Q}, & si\ 0\le X(k)<Q.\\ \dfrac{X(k)-Q}{0.5-Q}, & si\ Q\le X(k)<0.5.\\ \dfrac{1-Q-X(k)}{0.5-Q}, & si\ 0.5\le X(k)<(1-Q).\\ \dfrac{1-X(k)}{Q}, & si\ (1-Q)\le X(k)<1.\end{cases}$$

- *X(k)* étant l'entrée activatrice (Entrées d'un neurone ou 1 data-pixel de la clef utilisateur).
- *Q* étant le paramètre de contrôle.La fonction d'activation est dite dans un état chaotique si $0 < Q < 0.5$.
- Chaque fois que cette fonction d'activation s'exécute, on compte un tour de hachage de plus.

Les fonctions d'activation des neurones des différentes couches sont exprimées respectivement comme suit :

$$C=f^T\left(\begin{bmatrix}\sum_{i=0}^{3}(w_{0,i}*P_i)+b_{0,0}\\ \sum_{i=4}^{7}(w_{0,i}*P_i)+b_{0,1}\\ \cdots\\ \sum_{i=28}^{31}(w_{0,i}*P_i)+b_{0,7}\end{bmatrix},Q_0\right)=\begin{bmatrix}f^T\left(\sum_{i=0}^{3}(w_{0,i}*P_i)+b_{0,0},Q_0\right)\\ f^T\left(\sum_{i=4}^{7}(w_{0,i}*P_i)+b_{0,1},Q_0\right)\\ \cdots\\ f^T\left(\sum_{i=28}^{31}(w_{0,i}*P_i)+b_{0,7},Q_0\right)\end{bmatrix}=\begin{bmatrix}C_0\\ C_1\\ \cdots\\ C_7\end{bmatrix}$$

$$D=f(W_1C+B_1),Q_1)=\begin{bmatrix}f\left(\sum_{i=0}^{7}(w_{1,0,i}*C_i)+b_{1,0},Q_1\right)\\ f\left(\sum_{i=0}^{7}(w_{1,1,i}*C_i)+b_{1,1},Q_1\right)\\ \cdots\\ f(\sum_{i=0}^{7}(w_{1,7,i}*C_i)+b_{1,7},Q_1)\end{bmatrix}=\begin{bmatrix}D_0\\ D_1\\ \cdots\\ D_7\end{bmatrix}$$

$$H = f^T(W_2 D + B_2, Q_2) = \begin{bmatrix} f^T(\sum_{i=0}^{7}(w_{2,0,i} * D_i) + b_{2,0}, Q_2) \\ f^T(\sum_{i=0}^{7}(w_{2,1,i} * D_i) + b_{2,1}, Q_2) \\ f^T(\sum_{i=0}^{7}(w_{2,2,i} * D_i) + b_{2,2}, Q_2) \\ f^T(\sum_{i=0}^{7}(w_{2,3,i} * D_i) + b_{2,3}, Q_2) \end{bmatrix} = \begin{bmatrix} H_0 \\ H_1 \\ H_2 \\ H_3 \end{bmatrix}$$

- $f^T()$ est l'itération en T fois de la fonction d'activation $f()$, T doit être supérieur ou égal à 50 (pour garantir un résultat pseudo-aléatoire).
- Les paramètres de la fonction d'activation du NN-Hash rendent difficile (voir impossible) le calcul d'une fonction inverse. La somme pondérée à biais du premier paramètre permet une restitution des entrées d'un neurone inabordable. De plus, si on ne connait pas le deuxième paramètre de cette fonction Q, celui-ci ajoute une difficulté supplémentaire même en dehors de l'état chaotique. Alors, on peut dire que la fonction d'activation utilisée par le NN-Hash satisfait la condition du chiffrement à sens unique.

2.5.5. Modes de hachage :

Le NN-Hash utilise deux modes de hachage, le chiffrement en bloc, pour hacher directement des blocs de 1024 bits en 128bits ; et le chiffrement multi-bloc pour hacher des fichiers binaires de tailles arbitraires suite à un bourrage et une subdivision en blocs de 1024bits.

Pour réduire le nombre d'opérations exécutées sur les données, le NN-Hash peut utiliser un réseau de neurones parallèles[8] sans aucune incidence sur la complexité, la sécurité et la robustesse.

2.5.6. Sécurité :

[8]Réseau de neurone parallèle : Subdivision en sous-réseaux exécutés d'une façon parallèle d'un réseau de neurones quelconque afin d'accélérer le calcul.

Le NN-Hash, comme définie par son article, a un niveau de sécurité supérieur à 2^{64} bits pour des collisions simples, et de 2^{256} pour le brute force. Donc, cette fonction de hachage présente une sécurité optimale par rapport à la puissance de calculs d'aujourd'hui. On peut dire alors que le NN-Hash est une fonction de hachage complète.

3. Message AuthenticationCode:

Les MAC sont un moyen pour vérifier l'authenticité et l'intégrité d'un message donné. Son code est joint à un message pour détecter une fraude ou une altération quelconque.

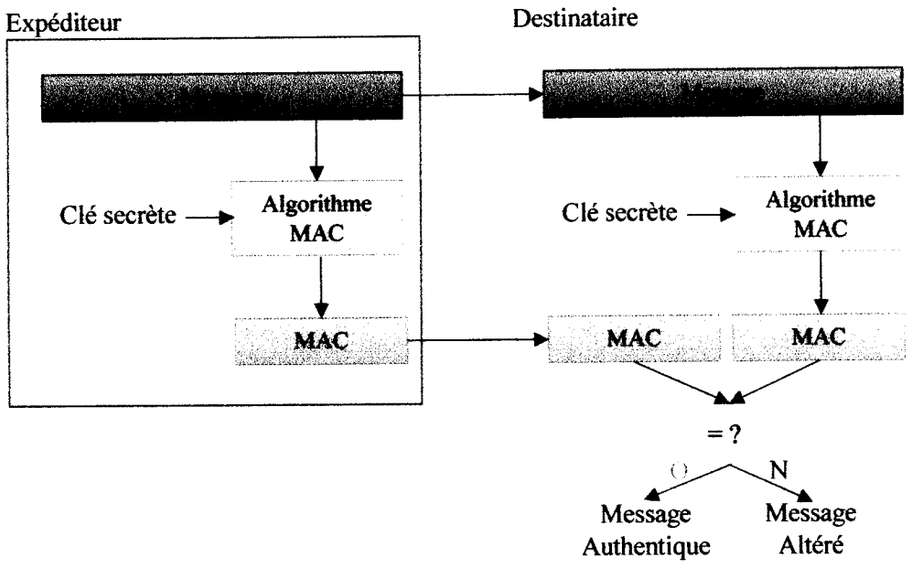


Figure 1-7. : Utilisation des MAC

Les différent algorithmes MAC cherchent à satisfaire, selon les besoins, une ou plusieurs contraintes dont :

- Temps de calcul faible.
- Taille de haché réduite.
- Taux de collision moindre : Sécurité optimale.
- Taux de collision exprimant une ressemblance des structures.

Il existe plusieurs constructions spécifiques pour les MAC, les plus connus sont :

Algorithme MAC	Principes de base
Checksum	Somme des octets.
CMAC	Algorithme de chiffrement, clé secrète.
HMAC	Algorithme de hachage, clé secrète.
MMH-Badger MAC	Somme modulaire (Message * clé).
Poly1305-AES	Algorithme de hachage AES, clé, nonce[9].
UMAC	Algorithme de hachage, algorithme de chiffrement.
VMAC	Algorithme de hachage, bourrage pseudo-aléatoire, clé.

Tableau 1-1. : Quelques mécanismes MAC

Pour ce Mémoire, on n'exposera que le HMAC (Hash-based Message Authentication Code).

3.1. Hash-based Message Authentication Code:

HMAC est un mécanisme de calcul des MAC, il incorpore une fonction de hachage (déjà existante) ainsi qu'une clé dans un schéma qui lui est propre. Les HMAC sont utilisés

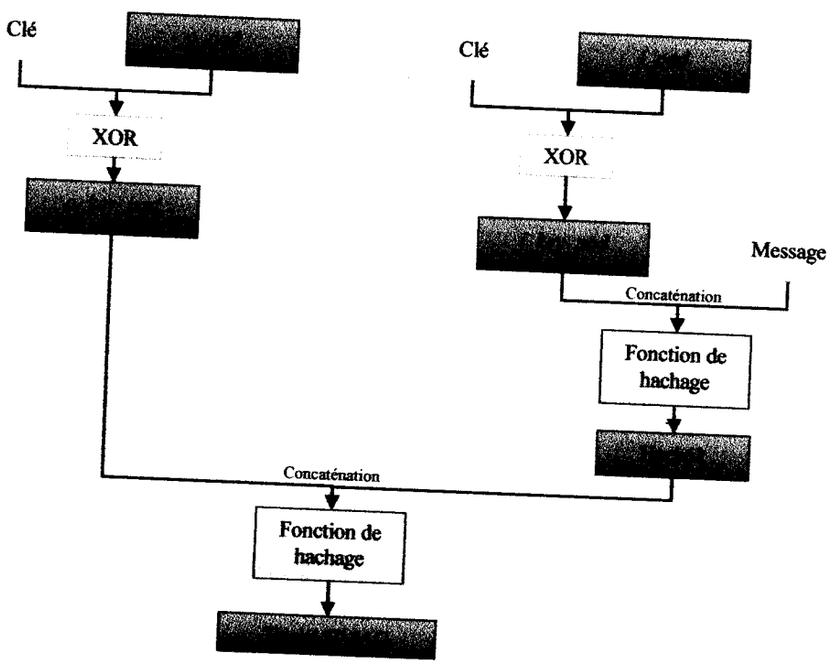


Figure 1-8. : Schéma général du HMAC

[9] Nombre aléatoire (utilisé qu'une seule fois) pour signer un ensemble de données.

pour à la fois l'intégrité et l'authenticité d'un message.

Remarque :

- La clé doit avoir la même taille que celle du bloc utilisé par la fonction de hachage choisie. (Si la taille de la clef est plus grande que celle du bloc, celle-ci est hachée)
- *o_pad* et *i_pad* sont des blocs (de la raille d'un bloc de la fonction de hachage) dont la valeur est respectivement : 0x5C5C…5C, 0x3636...36.
- La taille du haché HMAC est naturellement la même que celle du haché de la fonction de hachage choisie
- La sécurité d'un HMAC est intrinsèquement liée à celle de la fonction de hachage utilisée.

4. Attaques cryptanalytiques:

Il existe une grande panoplie d'attaques cryptanaytiques, nous présenterons que les attaques les plus connu visant, ou aidant à fragiliser les fonctions de hachage.

4.1. Le brute-force :

Est une attaque dont le principe est de tester toutes les possibilités de l'algorithme de cryptage pour aboutir à une restauration des données originales. La limite théorique de cette attaque n'a jamais dépassé une complexité de données de 2^{64}bits comme parcours moyen, c.-à-d. qu'une clé de 128bits ou plus ne peut être brute-forcée.

4.2. Attaque par paradoxe des anniversaires :

L'objectif de cette attaque est de trouver, dans un grand nombre de messages aléatoires, le même haché pour des couples de messages différents. L'attaque par paradoxe des anniversaires peut être très utile pour une attaque par seconde pré-image.

La résistance à cette attaque peut se calculer, dans le cas d'une simple fonction de hachage à approximativement $2^{N/2}$ (N étant la taille du haché).

La puissance de calcul d'aujourd'hui ne peut effectuer une telle attaque avec une résistance supérieure à 2^{64}.

4.3. Attaques par pré-image :

Est une attaque qui calcule depuis une empreinte généré par une fonction de hachage connue de l'attaquant, le message original. Il existe une autre attaque par pré-image qui se base sur les collisions, elle calcule un message différent dont le haché est le même que celui du message original, elle est appelé une attaque par seconde pré-image et elle a besoin de l'algorithme de hachage s'exécutant en moins en boite noire.

4.4. Résistance des fonctions de hachage aux attaques cryptographiques

Le tableau suivant montre la sensibilité de quelques fonction de hachage aux attaques cryptanalytiques les plus connues (résistance à l'attaque : +).

Fonction	Paradoxe des anniversaires	Pré-image	Seconde Pré-image
RIPEMD	-	+	+
RIPEMD-160	+	+	+
MD2	-	+	-
MD4	-	-	-
MD5	-	+	-
SHA-0	-	+	+
SHA-1	-	+	+
SHA-256	+	+	+
SHA-512	+	+	+
Tiger	-	+	-
GOST	-	-	-
NN-Hash	+	+	+

Tableau 1.2. : Quelques algorithmes de hachage et leurs résistances.

En plus des spécifications des fonctions de hachage, les contraintes de sécurité de ces dernières constituent un autre paramètre de choix. Par Exemple, la résistance à l'attaque par paradoxe des anniversaires s'impose dans le cas où la nature des données de départ n'est pas importante suite à l'utilisation de la fonction en question.

5. Conclusion :

Nous avons vu dans ce chapitre des caractériels entourant le hachage, l'authentification et les réseaux de neurones. Ces dernières nous ont poussés à imaginer un mécanisme d'authentification optimal et sécurisé permettant de satisfaire les contraintes de taille des hachés et de vitesse de calcul.

Nous reprenons dans le chapitre suivant le standard HMAC afin de proposer un nouveau schéma de calcul pour celui-ci. Nous profiterons de la flexibilité des réseaux de neurones pour modifier le NN-Hash pour la compression des hachés de la fonction utilisée par HMAC. Le choix de la fonction NN-Hash se justifie par sa souplesse qui peut manifester une capacité potentielle pour la satisfaction de nos objectifs. De plus, les qualités de l'algorithme SHA-256 en termes de performances de calcul et de sécurité nous encourage à l'implémenter pour un test relativement accompli.

Chapitre II

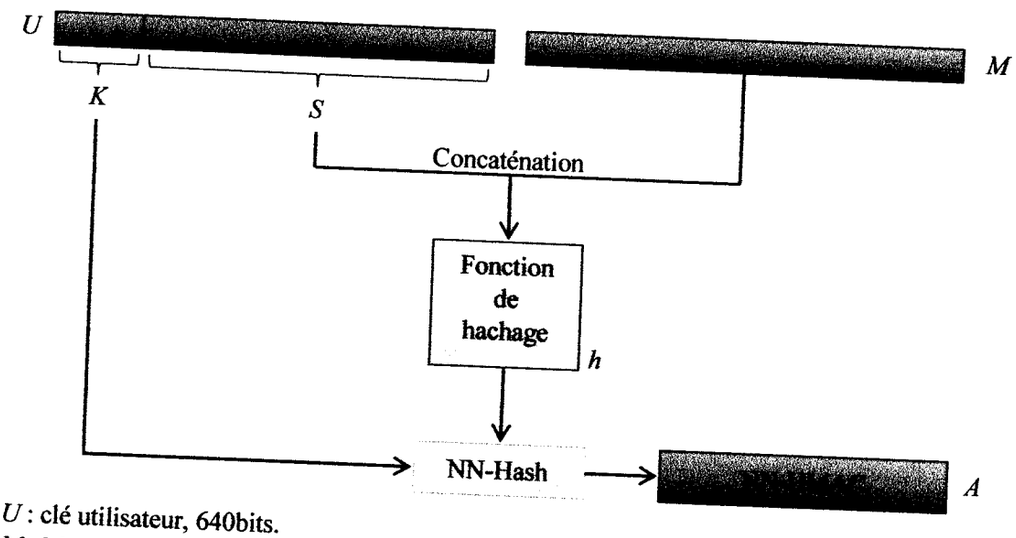
Vers un HMAC plus optimal

1. Introduction :

Nous spécifionsdans ce chapitre notre contribution pour l'optimisationdes HMAC (Hash-based Message Authentication Code). Pour cela, nous utiliserons le « One-way Hash FunctionBased on Neural Networks » (ou NN-Hash) au lieu du ré-hachage avec la même fonction qu'utilise un HMAC. On l'appellera NN-HMAC.

En remplaçant le deuxième hachage du HMAC par le NN-Hash, nous tentons d'améliorer HMAC.Une réductiondu nombre d'opération etdunombre de tours de hachage est attendue. Pour la démonstration de notre modèle, nous utiliserons SHA-256 comme fonction de hachage sous-jacente en vue de ses qualités de robustesse, performances et sécurité. Par la suite nous évaluerons les performances ainsi que la sécurité de ce modèle.

2. Schémas général :



U : clé utilisateur, 640bits.
M : Message à authentifier (taille quelconque).
K :clé NN-Hash, 128 bits.
S : clé de pré-hashage, 512bits.
A : Code d'authentification NN-HMAC, 32bits.

Figure 2-1. : Schéma général du modèle proposé.

Le modèle que nous proposons est simple. Il s'agit de calculer la somme de contrôle en deux temps. Nous utiliserons une clef de 640bits divisée en deux, une (512bits)

concaténée avec le texte à authentifier pour le pré-hachage en SHA-256, et l'autre (128bits) comme clé pour le NN-Hash. La taille du haché final de notre modèle sera de 32bits.

Le NN-HMAC est exprimé par :

$$NN\text{-}HMAC = NN\text{-}Hash(\, h(S \,||\, M),\, K)$$

Pour l'implémentation du NN-HMAC, nous pouvons utiliser n'importe quelle fonction de hachage (MD5, RIPEMD, SHA, ...), il suffit d'adapter le NN-Hash par rapport à la taille du haché de la fonction utilisée et celle du haché final. Dans ce mémoire, nous implémenterons SHA-256 pour comparer entre notre modèle et le HMAC-SHA256.

3. Adapter le NN-Hash pour le ré-hachage des signatures SHA256 :

Par défaut, le NN-Hash requière une entrée de 1024bits. Or, le haché d'un SHA256 est de, bien-entendu, 256bits. Pour des entrées de taille inférieures à 1024bits, le NN-Hash propose de compléter le message avec des bits «0» et un bit «1» en fin, et ça, avant la quantification.

Une réduction du nombre des neurones sur les différentes couches du réseau de neurones nous permet de mettre directement la signature SHA256 en entrée, gagner du temps-processeur pour le ré-hachage de la signature et d'arriver facilement à une sortie en un seul data-pixel au lieu de 4 (compression).

Naturellement, quelques modifications seront engendrées par la réduction du nombre de neurones. Les fonctions de transfert des différentes couches et le nombre de sous-clés seront reconsidérés par rapport à la structure de données des entrées du NN-Hash. Aucun autre aspect de ce système de hachage ne sera modifié.

Le réseau de neurones utilisé pour le ré-hachage des signatures SHA256 est une version modifiée du NN-Hash et il comprend :

- Une couche d'entrée :

  Composée de 2 neurones qui reçoivent8 data-pixels quantifiés, $P=[P_0,P_1,...,P_7]$ (signature SHA256), et qui transfèrent 2 data-pixels en sortie, $C=[C_0,C_1]$.

- Une couche cachée :

Composée de 2 neurones qui reçoivent les 2 data-pixels en sortie de la couche d'entrée, $C=[C_0,C_1]$, et qui transfèrent 2 data-pixels en sortie, $D=[D_0,D_1]$.

- Une couche de sortie :

Composée d'un seul neurone qui reçoit les 2 data-pixels en sortie de la couche cachée, $D=[D_0,D_1]$, et qui transfèrent un seul data-pixels en sortie final H, cette dernière est définie comme celle proposée par le NN-Hash :

$$H = f_2(W_2D + B_2) = f_2(W_2f_1(W_1C + B_1) + B_2) = f_2(W_2f_1(W_1f_0(W_0P + B_0) + B_1) + B_2)$$

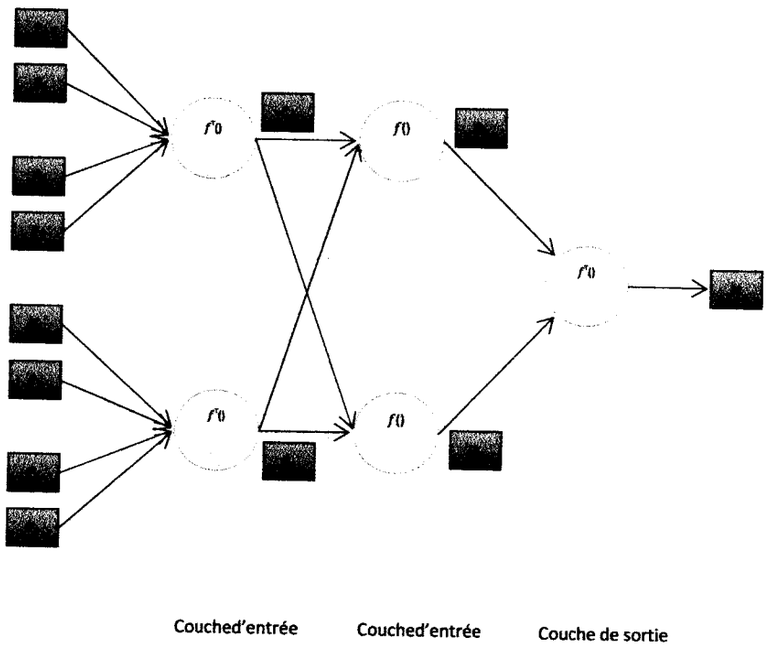


Figure 2-2. : Réseau de neurone du NN-Hash adapté.

3.1. Fonction générale de transfert :

La fonction de transfert générale reste la même pour le nouveau modèle :

$$f(X(k),Q) = \begin{cases} \dfrac{X(k)}{Q}, & si\ 0 \le X(k) < Q. \\ \dfrac{X(k)-Q}{0.5-Q}, & si\ Q \le X(k) < 0.5. \\ \dfrac{1-Q-X(k)}{0.5-Q}, & si\ 0.5 \le X(k) < (1-Q). \\ \dfrac{1-X(k)}{Q}, & si\ (1-Q) \le X(k) < 1. \end{cases}$$

3.2. Fonctions de transfert des différentes couches :

Couche d'entrée :

8 data-pixel en entrées pour 4 neurones dans la première couche nous poussent à faire les modifications suivantes :

- La matrice des sorties *C* est réduite de 8 sortie à 2.

$$C = f^T\left(\begin{bmatrix}\sum_{i=0}^{3}(w_{0,i} * P_i) + b_{0,0} \\ \sum_{i=4}^{7}(w_{0,i} * P_i) + b_{0,1}\end{bmatrix}, Q_0\right) = \begin{bmatrix} f^T(\sum_{i=0}^{3}(w_{0,i} * P_i) + b_{0,0}, Q_0) \\ f^T(\sum_{i=4}^{7}(w_{0,i} * P_i) + b_{0,1}, Q_0)\end{bmatrix} = \begin{bmatrix} C_0 \\ C_1 \end{bmatrix}$$

Couche Cachée :

2 neurones dans la deuxième couche nous poussent à faire les modifications suivantes :

- Chaque neurone prend 2 entrées *C* au lieu de 8.
- La matrice des sorties *D* est réduite de 8 sortie à 2.

$$D = f(W_1C + B_1), Q_1) = \begin{bmatrix} f(\sum_{i=0}^{1}(w_{1,0,i} * C_i) + b_{1,0}, Q_1) \\ f(\sum_{i=0}^{1}(w_{1,1,i} * C_i) + b_{1,1}, Q_1)\end{bmatrix} = \begin{bmatrix} D_0 \\ D_1 \end{bmatrix}$$

Couche de sortie :

Un seul neurone dans la troisième couche nous pousse à faire les modifications suivantes :

- Le neurone de sortie finale prend 2 entrées *D* au lieu de 8.
- La matrice des sorties *H* est réduite de 4 sortie à une seule.

$$H = f^T(W_2D + B_2, Q_2) = f^T(\sum_{i=0}^{1}(w_{2,i} * D_i) + b_2, Q_2)$$

3.3. Fonction de Hachage :

Il est, à priori, inutile de modifier la fonction de hachage, la taille de la clef utilisateur ou le générateur de clés :

$$\begin{cases} X_0(k) = f^{T+k}(K_0, K_1) \\ X_1(k) = f^{T+k}(K_2, K_3) \\ K_s(k) = \big(X_0(k) + X_1(k)\big) \bmod 1 \end{cases}$$

Le nombre de sous-clé $K_s(k)$, par contre, est réduit de 151 data-pixel à 22 data-pixel.

4. Analyse des performances :

Nous utiliserons, pour notre analyse des performances les valeurs suivantes : Soit M un message de 1024bits, *M= «Dynamiccryptographic hash functionsdifferfromtraditional hash functionsbecausetheyrequire a second parameter, securitylvl»* ; U une clé de 640bits, *U= «The securityparametercontrols the methodused&& the size of the output digest».*

L'Analyse de performances ce notre modèle se fait à travers le NN-HMAC-SHA256 en comparaison avec le HMAC-SHA256 :

	HMAC-SHA256(L, M)	NN-HMAC-SHA256(L, M, I-5o)
Taille du haché	256	32
Haché	*0x4F08D6513668E48494271882C0B581F8 6F5C68E37D526E2C014470F0D5A74170*	*0x1832CAA8*
Nombres d'opérations	13568	7483
Tours de hachage	512	305

Tableau 2-1. : Comparaison du HMAC-SHA-256 et du NN-HMAC-SHA-256.

4.1. Gains de l'optimisation du HMAC :

Pour quantifier notre objectif, on calcule les moyennes en hachant un ensemble de couples (Message, Clé) générés aléatoirement.

On a obtenu, sur n messages de 1024bits et n clés de 640 bits, les résultats suivants :

	Taille du haché	Nombre d'opérations	Tours de hachage
HMAC-SHA256	256	13568	512
NN-HMAC-SHA256	32	7380	305
Gain *(pourcentage)*	87.5 %	45.61 %	40.43 %

Tableau 2-2. : Gains d'optimisation du NN-HMAC-SHA256

Remarque : Le nombre d'opérations pour le calcul d'un HMAC ne change que si la taille du texte ou de la clé change. En revanche, le nombre d'opération peut changer pour le NN-HMAC suite à n'importe quelle modification du texte (taille et contenue). 7380 est une moyenne approximative extraite de la simulation du modèle.

4.2. Sensibilité du NN-HMAC :

Quelle est l'importance du changement sur le haché final causé par l'altération d'un seul bit du message ou de la clé ? Nous calculerons pour le savoir les distances de Hamming entre le haché du couple (M,U), le haché de (M'_i,U), et celui de (M,U'_i). M'_i et U'_i sont respectivement le message M et la clé U dont le i-ème bit a été modifié.

HDR ou Hamming Distance Ratio est le pourcentage de différence des bits entre deux chaines de caractères de même taille.

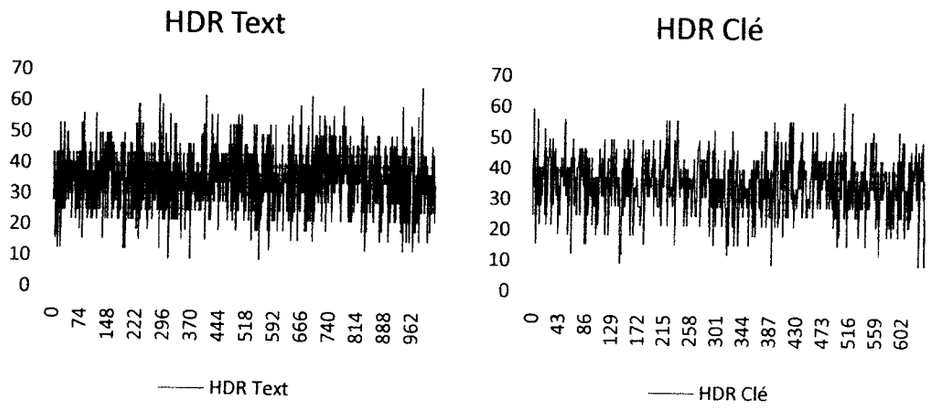


Figure 2-3. : Sensibilité du modèle proposé

On a extrait depuis les graphes de la *Figure 2-3.*, une moyenne approximative du HDR de 35% pour le changement d'un bit sur le message M (1024bits) et sur la clé U (640bits). On

peut dire donc que notre modèle a un changement considérable sur l'altération du couple (Message, Clé).

5. Analyse de sécurité :

5.1. Résistance à l'attaque par Brute-force :

Pour un HMAC dont la fonction de hachage est connue par l'attaquant, le brute-force tente de calculer la clé utilisée.

Doté d'une clé de 640bits, le NN-HMAC est résistant au brute-force, car celle-ci requière 2^{320} computations, valeur qui est nettement supérieure à la limite théorique de l'attaque.

5.2. Résistance à l'attaque par paradoxe des anniversaires :

Un haché d'une taille de 32bits peut générer une difficulté d'attaque par collision à 2^{16} qui la rend donc faisable. Or, pour le NN-HMAC, chaque clé (128bits) génère 2^{16} hachés, ce qui augmente la difficulté de trouver une collision cohérente à $2^{(16+64)}$. Alors, le NN-HMAC est résistant à l'attaque par paradoxe des anniversaire, elle est infaisable si la clef est inconnue.

5.3. Résistance à l'attaque par pré-image :

La résistance à l'attaque par pré-image dépend directement de la fonction de hachage utilisée par le NN-HMAC. Si la fonction en question est résistante à cette attaque, le NN-HMAC l'est surement, vu que le NN-Hash est, par défaut, résistant à cette dernière.

En conséquence de la compression du haché à 32bits, le NN-HMAC est sensible à l'attaque par seconde pré-image. En effet, on connaissant l'algorithme et la clé, un attaquant peut générer grâce à un message/haché authentiques, un autre message frauduleux ayant le même haché, et ainsi ce message frauduleux peut être détecté comme authentique par le destinataire.

6. Discussion et critiques :

Après l'étude des résultats obtenus, nous remarquons que notre modèle présente en plusieurs points des avantages et des inconvénients par rapport au HMAC.

L'avantage axiomatique, qui est le but principal de ce mémoire, est l'amélioration du calcul du HMAC. Evidemment, notre modèle s'exécute avec moins d'opération et de tours de hachage, ce qui rend la génération de son code rapide tout en ayant un résultat optimal en termes de taille. Grâce à un ratio performances/sécurité acceptable, notre modèle se propose d'élargir le champ d'utilisation du mécanisme HMAC. NN-HMAC peut donc être utilisé pour une authentification dans un environnement qui requière une complexité spatiale et calculatoire réduite (ex : intégrité des exécutables dans un système d'exploitation).

Pour un usage simple de l'authentification des messages, l'ajout du NN-Hash nous permet de garder une sécurité convenable comparé aux asservissements des modifications appliqués. Si les entités communicantes préservent leurs paramètres secrets, notre modèle est à priori fiable par rapport à la puissance de calcul d'aujourd'hui. De plus, l'adaptation du NN-Hash pour notre modèle masque parfaitement la fonction de hachage utilisée sans tronquer le haché, méthode qui amoindrie le niveau de sécurité de ce dernier.

Notre modèle possède une sensibilité très acceptable par rapport aux changements des entrées. Il peut aussi générer pour une seule clé plus de quatre milliard de hachés différents sur un total d'un peu moins de 2^{160} possibilités (2^{128} possibilités de clé * 2^{32} possibilités de haché).

Quant aux inconvénients, ils concernent tous la sécurité. Effectivement, et comme montré dans l'analyse de sécurité, notre modèle est vulnérable aux attaques de la seconde pré-image, ce qui le rend inutilisable dans un environnement où les paramètres de calcul sont attaquables… cela est valable peu importe la fonction utilisée.

Aussi, et malgré que notre modèle puisse, dans un contexte précis, être considéré comme une fonction de hash, il ne peut être employé, sans gestion des collisions, pour des usages tels que *«la table de hachage»* ou *«l'empreinte cryptographique»*; Pour ces cas de figure, la clé n'a plus d'importance, et par conséquent, le taux de collisions augmente.

7. Conclusion :

Nous avons présenté dans ce chapitre notre proposition, NN-HMAC, qui est un mécanisme d'authentification des messages, son objectif est d'améliorer HMAC. Une fonction de hachage basée sur les réseaux de neurones a été utilisée pour arriver à optimiser le standard en question, SHA-256 a été utilisé pour la simulation du modèle et pour la quantification de ses caractéristiques.

Les résultats de la simulation nous encouragent à comprendre plus le fonctionnement du mécanisme HMAC et de l'améliorer d'avantage. Malgré que notre modèle ait prouvé en certains points son efficacité, il reste néanmoins à revoir ou à compléter. Naturellement, une étude de sécurité approfondie valorisera notre modeste contribution et lui donneras le privilège d'être un mécanisme MAC assez complet.

Conclusion générale et perspectives :

Nous avons présenté dans ce mémoire une nouvelle approche pour les HMAC, on a optimisé ces derniers en intégrant une fonction de hachage basée sur les réseaux de neurones. La modification du schéma général nous a permis, entre autres, d'améliorer le temps de calcul (illustré par le nombre d'opérations et les tours de hachage).

Nous avons profité, pour la réalisation de ce travail, de l'efficience des réseaux de neurones pour la diffusion, compression et confusion de données. Nous avons, grâce au NN-Hash, conçu un *Hash-based Message Authentication Code*capable de générer des hachés de taille réduite, d'une assez bonne sensitivité et doté d'une sécurité, à priori, acceptable en un temps relativement inférieur.

Forcément, l'utilité d'études complémentaires s'impose pour valoriser et compléter le modèle que nous avons proposé.

Il serait avantageux de situer notre modèle dans une échelle de sécurité, une étude cryptanalytique pourrait définir d'une façon précise le bon environnement ainsi que les utilisations que NN-HMAC pourrait avoir.

Afin d'étudier notre modèle, on a utilisé l'algorithme de hachage SHA-256. L'étude de l'implémentation d'autres algorithmes de hachage dédiés peut satisfaire des contraintes d'ordre (niveau de sécurité/usage), et ainsi étendre le champ d'utilisation de ce mécanisme d'authentification.

Enfin, la réalisation de ce mécanisme dans un environnement réel serait intéressante et édifiante pour la compréhension de ce modèle par rapport aux modèles existants.

Bibliographie

[1] http://www.wikipedia.org/

[2] http://crypto.stackexchange.com/

[3] http://www.stackoverflow.com/

[4] http://preshing.com/20110504/hash-collision-probabilities/

[5] ShiguoLian, Jinsheng Sun, Zhiquan Wang, *One-way Hash Function Based on Neural Network*,Nanjing University of Science & Technology, China, 2007.

[6] Murali Krishna Reddy Danda, *Design and Analysis of Hash Functions*,Thèse de Master, Victoria University, Australia, 2007.

[7] Saif Mohammed S. A. Al-Kuwari, *Integrated-Key Cryptographic Hash Functions*,Thèse de doctorat, BATH University, United-Kingdom, 2011.

[8] FIPS 180-4, Federal Information Processing Standards Publication FIPS PUB 180-4, *Specifications for the Secure Hash Standard (SHS)*, March 2012.

[9] Krawczyk, H., Bellare, M., and R. Canetti, *"HMAC: Keyed-Hashing for Message Authentication"*, RFC 2104, 1997.

[10] M. Nystrom, *"Identifiers and Test Vectors for HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512"*, RFC 4231, 2005.