FingerKey, un cryptosystème biométrique pour l'authentification

Valérie Viet Triem Tong, Hervé Sibert, Jérémy Lecoeur, Marc Girault

FingerKey, un cryptosystème biométrique pour l'authentification

V. Viet Triem Tong, H. Sibert, J. Lecœur et M. Girault

Supelec, Campus de Rennes, Avenue de la Boulaie F-35576 Cesson-Sévigné Cedex, France
et NXP Semiconductors, 9 rue Maurice Trintignant, F-72081 Le Mans Cedex 9, France
et Irisa-Inria, Campus de Beaulieu, F-35042 Rennes Cedex
et France Telecom Research and Development, 42 rue des Coutures, BP6243, F-14066 Caen Cedex 4, France

Nous nous intéressons dans cet article à l'authentification des utilisateurs par le biais de leurs données biométriques (empreinte digitale, forme de la main, ...). Traditionnellement, l'authentification biométrique d'un utilisateur consiste à vérifier que sa donnée biométrique courante est suffisamment proche d'une donnée de référence. Malheureusement, la sécurité de ce schéma souffre du fait que les données biométriques sont des données personnelles non révocables. Lorsqu'une donnée biométrique est compromise, contrairement à un mot de passe, elle ne peut pas être changée. Nous pensons que le point faible des approches traditionnelles réside dans le stockage des données biométriques de référence. Si les données biométriques n'étaient pas stockées, elles seraient plus difficiles à voler. Il serait aussi plus difficile d'en compromettre un grand nombre simultanément. Pour pallier ce problème, nous proposons un schéma d'authentification biométrique ne nécessitant pas la comparaison à une valeur biométrique de référence. Notre méthode améliore la sécurité de l'authentification biométrique puisqu'elle ne nécessite pas de stockage.

Mots-clés: cryptographie, biométrie, authentification

1 Motivations

Force est de constater que la biométrie s'impose un peu partout : nous disposons de capteurs d'empreinte digitale pour nous authentifier envers nos ordinateurs ou nos clés USB, l'accès à certains lieux (cantines, aéroports, entreprises) est contrôlé grâce à une vérification biométrique des usagers autorisés. L'engouement pour la biométrie s'explique par le fait qu'elle offre une manière ergonomique d'authentifier ou d'identifier un utilisateur.

De façon générale, l'authentification biométrique consiste à vérifier que la donnée biométrique courante de l'utilisateur est suffisamment proche d'une donnée de référence enregistrée précédemment. Cette méthode d'authentification est ergonomique, puisqu'elle ne nécessite pas que l'utilisateur se souvienne d'un mot de passe, il suffit qu'il pose son doigt (sa main, ...) sur un capteur. Néanmoins ce schéma n'est pas satisfaisant du point de vue de la sécurité. En effet, les données biométriques d'un même utilisateur sont limitées et ne sont pas révocables. Baser l'authentification sur la biométrie revient donc à utiliser un mot de passe qui soit :

- toujours le même pour une application donnée
- toujours le même pour toutes les applications

Par ailleurs, et cela n'améliore en rien l'aspect sécurité, les données biométriques peuvent être stockées de manière centralisée. Si une base de données biométriques était compromise (contenant par exemple des empreintes), cela signifierait que tous les utilisateurs enregistrés ne devraient plus utiliser leurs empreintes digitales pour s'authentifier.

Pour répondre à ce problème, Uludag *et al* avancent dans [UPPJ04], l'idée qu'il serait préférable d'authentifier les utilisateurs en comparant une donnée secrète indépendante mais calculée à partir de la donnée biométrique. La sécurité du schéma d'authentification biométrique serait améliorée

puisqu'il ne nécessiterait pas de stocker les données biométriques de référence. A notre connaissance, il n'existe aujourd'hui aucune méthode d'authentification qui soit utilisable en pratique et qui ne permettent pas de retrouver des informations sur les données biométriques utilisée. Nous proposons une telle méthode pour l'empreinte digitale dans cet article. Notre travail s'inspire de celui de Juels et Wattenberg [JW99] et s'appuie sur une caractérisation de l'empreinte par analyse de texture introduite en 2000 par A.K. Jain et S. Prabhakar dans [Pra01, JPHP00]. Dans ce qui suit, nous présenterons tout d'abord deux caractérisations d'une empreinte digitale (section 2), ensuite nous présenterons l'état de l'art des méthodes permettant l'authentification par biométrie sans stockage de données biométriques (section 3), ensuite nous présenterons rapidement les techniques de corrections d'erreurs utilisées (section 4), avant de présenter notre approche (section 5), l'étude de la sécurité de l'algorithme (section 6) et ses applications possibles avant de conclure par le bilan de notre travail (section 7).

2 Caractérisation d'une empreinte digitale

L'empreinte digitale est le modèle du relief cutané des doigts, ce relief se forme aléatoirement durant la période fœtale. Son caractère aléatoire s'affranchit des risques de ressemblance entre individus ayant le même patrimoine génétique. La probabilité pour que deux personnes aient la même empreinte digitale est très faible, elle est d'environ 1 chance sur 64 milliards. La caractérisation de l'empreinte digitale d'un individu offre donc un moyen d'authentifier cet individu.

La classification des empreintes peut tout d'abord se baser sur la topographie générale de l'empreinte suivant le système de Henri. Dans ce système, la classification permet de définir des familles d'empreintes telles que les boucles, les arches, les tourbillons. Cette classification est trop générale n'est pas assez performante pour être utilisé sur un grand nombre d'empreintes. Nous présentons dans ce qui suit la caractérisation de l'empreinte par localisation des points caractéristiques appelés minuties. Cette caractérisation est la plus utilisée. Nous présentons aussi la caractérisation par analyse de texture que nous utiliserons dans ce papier.

2.1 Caractérisation par localisation des minuties

Les éléments qui permettent de différencier deux empreintes digitales sont les points caractériques qui sont appelés minuties. Une minutie est un point qui se situe sur le changement de continuité des lignes papillaires. L'approche la plus utilisée pour comparer deux empreintes se fait en comparant la localisation des minuties. Nous ne donnerons pas plus de détails sur les algorithmes de comparaison car ils ne sont pas l'objet de cette étude. Nous renvoyons notre lecteurs vers d'autres sources comme par exemple [JM03]. Nous expliquons seulement ce qu'il faut retenir de cette caractérisation pour le travail que nous présentons.

Le problème de la caractérisation de l'empreinte par localisation des minuties est qu'il n'est pas possible de donner une représentation stable en taille et ordonnée des positions des minuties. Les images acquises sur les capteurs biométriques sont soumises à d'importantes variations, à cause de traumatismes, température, ou vieillissement. De ce fait les minuties extraites peuvent disparaître d'une image à une autre ou ne pas être localisée exactement de la même manière. Pour notre travail, nous avons préféré pouvoir disposer d'une représentation stable en taille et ordonnée, comme cela sera expliqué ultérieurement. Nous présentons maintenant la caractérisation par analyse de texture qui présente cet avantage.

2.2 Caractérisation par analyse de texture

Jain et Prabhakar ont proposé dans [Pra01, JPHP00, JRP01] une méthode de caractérisation de l'empreinte par analyse de texture. De façon générale, cette méthode se focalise sur la mise en valeur des crêtes et vallées de l'empreinte autour d'un point fixe appelé centre morphologique. Dans cette approche, une empreinte est caractérisée par un vecteur de 640 composantes appelé FingerCode. Chaque composante du vecteur est calculée à partir des valeurs de gris dans des secteurs autour du centre morphologique. La comparaison entre deux empreintes se fait par le calcul de la distance

euclidienne entre les deux FingerCodes. Nous détaillons ici le traitement de l'image d'une empreinte permettant le calcul d'un Fingercode.

La première étape du travail consiste à retrouver le centre morphologique de l'image qui le point de plus forte courbure positive. Il se trouve donc en calculant un estimateur de courbure pour chaque pixel de l'image. Dans l'étude que nous avons mené sur des images $500 - ppp$, le centre morphologique calculé est exact dans 83% des cas et éloigné d'au plus 4 pixels du centre réel dans 14% des cas, donc exploitable dans 97% des cas.

Une fois le centre morphologique calculé, nous extrayons une zone circulaire autour de ce point. Cette zone est découpée suivant un pavage circulaire constitué de 5 bandes concentriques et de 16 secteurs angulaires, ce découpage donne un total de 80 secteurs. La largeur d'une d'une bande dépend de la qualité de l'image ; pour une image de résolution $500 - ppp$, la largueur d'une bande est de 20 pixels. Le fait de restreindre l'étude de l'empreinte à une zone circulaire autour d'un centre morphologique permet de s'abstraire des problèmes de rotation et de translation.

Les crètes et les vallées de l'image sont caractérisées par leur fréquence locale et leur orientation. En utilisant des filtres de Gabor [Gab46] bien choisis, il est possible d'en extraire les caractéristiques. En effet, lorsque ceux-ci sont correctement paramétrés, ils permettent d'enlever le bruit, de préserver les structures des crètes et des vallées et fournissent des informations sur l'orientation locale du motif. Nous caractérisons nos empreintes suivant 8 directions.

Pour finir, nous calculons pour chacun des 80 secteurs et pour chacune des 8 directions, l'écart moyen absolu à la moyenne, ce qui constituera notre vecteur de caractéristiques.

Les valeurs 0 à 79 du FingerCode correspondent aux valeurs obtenues après l'application d'un filtre de Gabor de degré 0, les 80 valeurs suivantes correspondent aux valeurs obtenues avec un filtre orienté à $22,5^{\circ}$, etc. Pour chaque série de 80 valeurs, les 16 premières sont celles obtenues pour la bande la plus interne. Pour comparer deux FingerCodes, il suffit de calculer la distance euclidienne entre les FingerCodes. La rotation du doigt sur le capteur se traduit par la rotation des valeurs sur le FingerCode. Ainsi, une simple rotation cyclique des valeurs du FingerCode permet de gérer des rotations du doigt, *e.g.* une rotation cyclique de deux crans compense une rotation du doigt de 45°.

Nous avons préféré utiliser la caractérisation de l'empreinte par son FingerCode car elle présente l'avantage d'offrir une représentation stable en taille (640 composantes) et ordonnée (les composantes d'un FingerCode sont comparables deux à deux).

3 Etat de l'art

Il n'a été proposé que deux méthodes d'authentifications permettant l'authentification biométrique dans stockage de données biométriques. Dodis *et al* donnent le nom général de *Fuzzy extractor* à ces méthodes dans [DRS04]. Le terme anglais *fuzzy*, littéralement *flou*, rappelle l'idée que l'authentification biométrique n'est pas un procédé exact puisque les données biométrique sont, par nature, variables. Les perfomances d'un système biométrique se mesurent à l'aide d'un pourcentage de fausses acceptations mis en correspondance avec un taux de faux rejets. Il n'a été proposé que deux *Fuzzy extractors* jusqu'à présent. Nous expliquerons que le premier n'est pas exploitable en pratique sur des empreintes digitales. Nous expliquerons aussi pourquoi le second n'est pas entièrement satisfaisant.

3.1 Fuzzy commitment

L'idée générale de notre étude est de s'intéresser aux moyens de s'abstraire de la comparaison à une donnée biométrique de référence. En 1999, Juels et Wattenberg sont les premiers à proposer dans [JW99] une solution appelée *Fuzzy commitment* , litéralement *engagement flou* . Le principe de leur méthode est d'utiliser la donnée biométrique courante pour recalculer une valeur qui servira ensuite pour l'authentification de l'utilisateur. Plus formellement, la méthode dite du *fuzzy commitment* se décompose en deux étapes : enrôlement puis authentification. Traditionnellement en biométrie, l'étape d'enrôlement consiste à relever plusieurs fois la donnée biométrique

de l'utilisateur (son empreinte par exemple) afin de constituer une valeur de référence. L'étape d'authentification consiste ensuite à comparer la valeur courante à la valeur de référence pour déterminer si l'utilisateur est bien celui attendu. La méthode dite du *fuzzy commitment* suit ces deux étapes sauf que la valeur de référence stockée n'est pas une donnée biométrique et ne permet pas de retrouver la donnée biométrique utilisée pour la générer. Pour cela, le *fuzzy commitment* utilise un ensemble de mots de code dans $\{0,1\}^n$ et une fonction de hachage $\mathcal{H}$.

Enrôlement : Un utilisateur U présente son empreinte biométrique représentée sous la forme d'une suite x de n bits. Le système choisit aléatoirement un mot de code $c \in \{0,1\}^n$ et calcule $(c-x, \mathcal{H}(c))$. Le *fuzzy commmitment* de l'utilisateur U est le couple $(c-x, \mathcal{H}(c))$. Cette donnée est stockée, le reste est effacé. A la fin de cette étape, le système authentifiera comme étant U tout utilisateur capable de produire une empreinte biométrique permettant de retrouver le mot de code c.

Authentification : Supposons qu'un utilisateur se présente sous l'identité de U et que son signal biométrique courant soit x'. Le système utilise x' pour vérifier l'engagement $(c-x, \mathcal{H}(c))$. Pour cela, il faut tout d'abord calculer $(c-x)+x'$. Si l'utilisateur était bien celui qu'il prétend être, alors son empreinte biométrique courante x' devrait être proche de son empreinte biométrique de référence x et par voie de conséquence $(c-x)+x'$ devrait être proche de c, au sens de la métrique de Hamming. Si la distance entre $(c-x)+x'$ et c est inférieure à la distance du code, alors le mot de code c', le plus proche de $(c-x)+x'$ est égal à c. Pour cela, il suffit de vérifier que $\mathcal{H}(c') = \mathcal{H}(c)$ où c' est le mot de code le plus proche de $(c-x)+x'$.

L'idée de Juels et Wattenberg apporte, en théorie, une solution au problème que nous nous posons. Malheureusement, leur approche repose sur l'hypothèse qu'une donnée biométrique dispose d'une représentation binaire canonique, stable en taille et ordonnée. Ce qui n'est pas le cas en pratique pour l'empreinte digitale lorsque l'on utilise la caractérisation habituelle basée sur la localisation des minuties. La représentation n'est pas stable en taille puisqu'une minutie peut apparaître ou disparaître suivant la qualité de la capture. La représentation n'est pas non plus ordonnable, pour la même raison. Notre première idée a été d'utiliser la caractérisation de l'empreinte par analyse de texture dans l'approche de Juels et Wattenberg. Mais notre expérimentation a montré trop de variations dans les FingerCodes d'une même personne pour que cela soit utilisable en pratique.

3.2 Fuzzy vault

Juels et Sudan ont proposé une autre approche appelée *fuzzy vault* dans [JS02]. L'authentification d'un utilisateur repose, là encore, sur sa capacité à présenter une donnée biométrique permettant de retrouver un secret. Dans [JS02], le secret en question est un polynôme P de degré n. Lors de l'étape d'enrôlement, le système calcule un *fuzzy vault* $\mathcal{V}$, littéralement *coffre fort flou*, à partir de P et de l'ensemble des minuties de l'empreinte de l'utilisateur. L'utilisateur est authentifié lorsqu'il est possible de retrouver P à partir de $\mathcal{V}$ et des minuties apparaissant dans son empreinte courante. Plus précisément, l'algorithme de Juels et Sudan peut se présenter comme suit :

Enrôlement : Considérons un utilisateur U, l'ensemble des minuties $\{a_i\}_{1\leq i\leq t}$ apparaissant dans l'image de son empreinte digitale et un polynôme P de degré k où $t > k$. Nous calculons tout d'abord l'ensemble des points $R_1 = \{(a_i, P(a_i)\}_{1\leq i\leq t}$. Nous générons ensuite un second ensemble de points $R_2 = \{(x_j, y_j\}_{1\leq j\leq n}$ tels que $x_j \neq a_i$ et $y_j \neq P(a_i)$ pour tout $1 \leq i \leq t$ et tout $1 \leq j \leq n$. L'ensemble R_2 est destiné à brouiller l'information venant de R_1. Le *coffre-fort flou* de l'utilisateur U est l'ensemble $\mathcal{V}_U = R_1 \cup R_2$. Cet ensemble est rendu publique, le reste est effacé.

Authentification : Lorsqu'un utilisateur souhaite s'authentifier en temps que U, il présente son empreinte digitale et nous extrayons l'ensemble des minuties $\{b_i\}_{1\leq i\leq t'}$ de l'image acquise. Nous sélectionnons ensuite dans $\mathcal{V}_U$ l'ensemble des points de la forme (x,y) où x est proche d'une valeur b_i. Si l'empreinte proposée est suffisament proche de l'empreinte de l'utilisateur U alors ce procédé

permet de retrouver suffisamment de points de R_1 pour pouvoir recontruire P par interpolation de Lagrange. Si c'est le cas, l'utilisateur est authentifié en tant que U.

L'approche du *fuzzy vault* présente l'avantage d'être utilisable en pratique. Clancy et Dennis ont présenté dans [CKL03] de bons résultats pratiques en prenant P un polynôme de degré 14 dans $\mathbb{F}_q[X]$ où $q = 251^2$. Ils proposent de publier un ensemble $\mathcal{V}_U$ de 313 points contenant entre 25 et 60 vrais points. L'étude de Clancy et Dennis s'appuie sur une représentation pré-alignée des minuties, dans Uludag et Jain présentent dans [UJ04] une étude similaire dans laquelle les minuties ne sont pas pré-alignées.

Le problème de cette seconde méthode est qu'elle nécessite de publier un ensemble de points dans lequel certaines abscisses correspondent à des minuties. C'est à notre avis la faiblesse de cette approche. Si cette approche était utilisée, il faudrait, en pratique, renouveler régulièrement l'ensemble des points publiés. L'ensemble des vrais points se définirait alors simplement par l'intersection de tous les ensembles publiés. Par ailleurs, Chang et Li ont étudié dans [CL06] le problème général du calcul d'un ensemble de valeur R_2 destiné à masquer un autre ensemble R_1. Leur travail a montré que le calcul de R_2 est difficile dès que la distribution des points de R_1 n'est pas uniforme.

Nous proposons une autre approche que nous appelons **FingerKey**. Notre approche s'inspire de la méthode du *fuzzy commitment* de Juels et Wattenberg, et utilise la caractérisation de l'empreinte par analyse de texture. Nous avons réalisé une expérimentation de notre approche qui montre qu'elle est utilisable en pratique. Nous montrerons que notre approche présente l'avantage par rapport à celle du *fuzzy vault* de ne pas publier des données permettant pas de recalculer les données biométriques de l'utilisateur. Notre construction utilise des codes correcteurs d'erreurs pour lesquels nous proposons quelques rappels dans la section suivante.

4 Notions de codes correcteurs

Nous ne présentons ici que les notions de codages utiles à la compréhension de notre schéma. Pour plus de détails, nous renvoyons notre lecteur à [MS77a, MS77b].

Les codes correcteurs d'erreurs sont utilisés de façon à permettre la correction des erreurs de transmission d'un message sur un canal de communication. Les codes reposent sur l'ajout d'information redondante au message avant sa transmission. La redondance d'information permet de détecter la présence d'erreurs si le message reçu est altéré (codes détecteurs), voire de reconstruire le message original (codes correcteurs). Dans notre schéma, nous utiliserons les codes correcteurs pour atténuer les variations du signal biométrique.

Formellement, un code correcteur d'erreur sur un espace de message $M = \{0,1\}^k$ consiste en un ensemble de mots de code $C \subset \{0,1\}^n$ où $n > k$. L'ensemble C est associé à une fonction d'encodage que nous noterons `encode` et à une fonction de décodage que nous noterons `decode`. La fonction `encode` est une injection $M \rightarrow C$ qui associe à chaque mot m de k bits de M un unique mot de code c de C. La fonction `decode` est une application de $\{0,1\}^n \rightarrow C \cup \emptyset$ qui fait correspondre à chaque mot m de n bits le mot de code le plus proche de m quand cela est possible et $\emptyset$ sinon. La métrique utilisée est la distance de Hamming selon laquelle la distance entre deux chaînes de bits de même longueur est le nombre de bits dont elles diffèrent. La distance d'un code est la plus petite distance entre 2 mots de codes. Pour qu'un mot de n-bits puisse être décodé, il faut qu'il soit à une distance inférieure à $\frac{d-1}{2}$ d'un mot de code et donc avoir au plus $\frac{d-1}{2}$ erreurs. . Le pouvoir de correction d'un code est la distance minimale à un mot de code pour qu'un message puisse être décodé. Dans notre expérimentation, nous avons choisi d'utiliser les codes de Reed Solomon pour leur pouvoir de correction.

5 Approche proposée

Notre approche se divise, comme usuellement en biométrie, en deux étapes : enrôlement et authentification. Lors de l'étape d'enrôlement, un utilisateur U présente son empreinte digitale, le

Algorithme FingerKey: Fuzzy Extractor pour un utilisateur U
Variable
  p : *polynôme* { *degré* k }
  n : *entier* { $n > k$ }
  $p_i = (x_i, p(xi))$: *point de* p { $0 \leq i \leq n$, *choisi aléatoirement* }
  $F_U = f_0 || \ldots || f_{n-1}$: *FingerCode*
Début
  K_{F_U} := $\emptyset$
  Pour i **variantDe** 0 **à** n-1 **Faire**
    c_i := RS-encode $(x_i, p(xi))$
    δ_i := $c_i - f_i$
    K_{F_U} := $K_{F_U} || (\delta_i, \mathcal{H}(c_i))$
  FinPour
  { *FingerKey* }
  Retourner K_{F_U}
Fin

Fig. 1: Algorithme d'enrôlement

système génère aléatoirement un polynôme secret p pour U. Nous utilisons le FingerCode de U et le polynôme p pour créer le **FingerKey** de U. Le FingerCode et p sont ensuite effacés. Supposons maintenant qu'un utilisateur souhaite s'authentifier et qu'il prétende être l'utilisateur U. Nous utilisons le **FingerKey** de U et l'empreinte courante de l'utilisateur pour essayer de retrouver p. Si le calcul termine sans échec alors l'utilisateur est authentifié en tant que U, sinon il est rejeté. Notre approche utilise la caractérisation de l'empreinte par le FingerCode, elle utilise aussi des codes correcteurs d'erreurs et l'interpolation de Lagrange. La première sous-section (5.1) qui suit présente l'étape d'encodage du FingerKey, qui correspond à l'étape d'enrôlement. La sous-section suivante (5.2) présente l'étape de décodage qui correspond à l'étape d'authentification.

5.1 Enrôlement

L'étape d'enrôlement d'un utilisateur U consiste à calculer son **FingerKey**. Pour cela, nous utilisons le FingerCode obtenu à partir de l'empreinte de U, que nous notons F_U. Ce FingerCode est un vecteur de 640 composantes, chaque composante est un nombre compris entre 0 et 7. Nous partageons F_U en n parties égales, ce que nous notons $F_U = f_0 || \ldots || f_{n-1}$. Nous générons ensuite aléatoirement un polynôme p de degré k où $k < n$. Sur ce polynôme, nous choisissons aléatoirement n points $p_i = (x_i, p(xi))$ pour $0 \leq i \leq n$. Chacun des n points p_i va être encodé à l'aide de la fonction d'encodage d'un code correcteur. Dans notre expérimentation, le code choisi est un code de Reed-Solomon, nous notons `RS-encode` la fonction d'encodage. La fonction de décodage sera, de la même manière, notée `RS-decode`. L'encodage des n points du polynôme permet d'obtenir n mots de code que nous notons $c_i = \texttt{RS-encode}(p_i)$ pour $0 \leq i \leq n$. Le **FingerKey** K_{F_U} de l'utilisateur U se définit ensuite comme la concaténation des n couples $(\delta_i, \mathcal{H}(c_i))$ où $\delta_i = (c_i - f_i)$, $\mathcal{H}$ est une fonction de hachage et $0 \leq i \leq n$. La figure 4. propose une version synthétique de l'algorithme d'enrôlement.

5.2 Authentification

Nous décrivons l'authentification d'un utilisateur prétendant être l'utilisateur U. L'authentification de cet utilisateur dépend de sa capacité à retrouver le polynôme secret p ayant servi à calculer le **FingerKey** K_{F_U} de U. Nous décrivons ici les étapes de l'algorithme présenté sur la figure 5.

Pour commencer, nous calculons le FingerCode courant F' de l'utilisateur. Si l'utilisateur est bien celui qu'il prétend être alors son FingerCode courant devrait être proche du FingerCode F_U qui a servi à calculer p. Nous partageons F' en n parties, comme nous l'avions fait pour F_U. Nous notons $F' = f'_0 || \ldots || f'_{n-1}$. Nous utilisons ensuite le **FingerKey** K_{F_U} de l'utilisateur U et effectuons

Algorithme Secure sketch pour un utilisateur prétendant être U
Variable
  k : *entier* { *degré du polynôme à retrouver* }
  n : *entier* { $n > k$ }
  $F'_U = f'_0 \oplus \ldots \oplus f'_{n-1}$: *Fingercode de l'utilisateur*
  $K_{F'_U} = (\delta_0, \mathcal{H}(c_0)) \oplus \ldots \oplus (\delta_{n-1}, \mathcal{H}(c_{n-1}))$: *Fingerkey de* U
Début
  P := $\emptyset$
  Pour i **variantDe** 0 **à** n-1 **Faire**
    c'_i := RS-decode $(\delta_i + f_i)$
    Si $\mathcal{H}(c_i) = \mathcal{H}(c'_i)$ **Alors**
      $P := P \cup \{c_i\}$
    FinSi
  FinPour
  Si $|P| \geq k$ **Alors**
    Retourner $p :=$ Interpolation de Lagrange des élements de P
  Sinon
    Retourner ECHEC
  FinSi
Fin

Fig. 2: Algorithme de décodage

les additions $f'_i + \delta_i$ pour $0 \leq i \leq n$. Si le FingerCode courant F' est proche de F_U alors les valeurs $f'_i + \delta_i$ sont proches des valeurs c_i. La proximité s'entend ici au sens de la métrique de Hamming. Comme les valeurs c_i sont des mots de codes, l'application de la fonction de décodage permet de retrouver les valeurs c_i, si les valeurs $f'_i + \delta_i$ sont à une distance inférieure à la capacité de correction du code choisi. Si le décodage est possible alors RS-decode$(f'_i + \delta_i)$ devrait permettre de retrouver c_i. L'ensemble des empreintes $\mathcal{H}(c_i)$ servent de témoins pour vérifier que l'utilisateur a su retrouver les valeurs c_i. Pour terminer, nous utilisons le fait que les c_i ont été encodés à partir de points aléatoires sur le polynôme. Si le FingerCode de l'utilisateur ne permet pas de retrouver au moins $k+1$ valeurs c_i parmis les n alors l'authentification échoue. Dans le cas contraire, nous disposons de $k+1$ points sur p et comme p est de degré k, nous utilisons l'interpolation de Lagrange pour retrouver p. L'utilisateur est authentifié comme étant U.

Dans ce qui suit, nous discutons de la sécurité de ce schéma et présentons l'expérimentation que nous avons menée.

5.3 Expérimentation

Nous avons réalisé un prototype de l'approche que nous venons de présenter, afin d'évaluer sa pertinence sur des données biométriques réelles. Nous avons pu disposer d'environ 1000 empreintes digitales. Pour notre expérience, nous avons arrondi les valeurs trouvées dans le calcul du FingerCode. Les valeurs comprises entre 0 et 2 ont été ramenées à 0, celles comprises entre 2 et 4 ont été ramenées à 1, les autres à 3. Après cet arrondi, il reste 3^{640} choix possibles pour un FingerCode. Nous avons divisé le FingerCode en $n = 16$ parties f_i. Notre expérience a montré que si nous ne tolérons aucun utilisateur authentifié à tort alors nous pouvons choisir un polynôme secret de degré 8. Notre expérience a montré qu'alors, un utilisateur légitime était authentifié dans 60% des cas.

6 Sécurité de l'approche proposée

Nous avons présenté un schéma d'authentification biométrique dans lequel les données biométriques ne sont pas stockées : il n'y a pas de comparaison à une valeur biométrique de référence.

Vérifier la sécurité de notre approche consiste a vérifier que les **FingerKey** publiés ne permettent de retrouver ni les données biométriques ayant servies à leur calcul ni le polynôme secret p autrement qu'en utilisant un FingerCode proche de celui utilisé lors de l'enrôlement. Nous montrons dans le théorème suivant que la complexité du calcul permettant de retrouver p est au mieux équivalente à celle permettant d'inverser la fonction de hachage. La sécurité de notre schéma est donc assurée tant que la fonction de hachage n'est pas inversible.

Théorème 1 *Considérons les paramètres suivants : n le nombre de division f_i faites dans le FingerCode, l la longueur d'une division f_i, d la distance du code correcteur utilisé, k le degré du polynôme secret p, L la longueur d'une empreinte pour la fonction de hachage $\mathcal{H}$ utilisée, $\mathcal{C}(\mathcal{H})$ la complexité de l'inversion de la fonction de hachage.*

Etant donné un **FingerKey** *K_f, la complexité du calcul permettant de retrouver p est égale à*

$$\min(2^L, (k+1) * \mathcal{C}(\mathcal{H}), (k+1) * \frac{2^\ell}{\binom{\ell}{\frac{d-1}{2}}})$$

Preuve Nous faisons ici l'hypothèse que le polynôme p est choisi dans un ensemble de polynômes suffisamment grands pour ne pas permettre la recherche exhaustive. Il est clair qu'un attaquant est capable de retrouver le polynôme p si et seulement si il connaît au moins $k+1$ points sur p. En l'absence d'informations supplémentaires, cela signifie qu'il a su retrouver $k+1$ éléments dans $\{c_i \in C\}_{1 \leq i \leq n}$, puisque les valeurs δ_i ne révèlent pas d'informations sur les c_i. L'attaquant peut choisir de chercher une valeur f tel que $\mathcal{H}(\texttt{RS-decode}(\delta_i + f)) = \mathcal{H}(c_i)$ pour $k+1$ différentes valeurs de i. Il peut aussi choisir d'inverser la fonction de hachage $\mathcal{H}$.

S'il opte pour la première solution, il doit trouver des valeurs f telles que la distance entre c_i et $\delta_i + f$ soit au plus $\frac{d-1}{2}$. Il doit alors tester en moyenne $\frac{2^\ell}{\binom{\ell}{\frac{d-1}{2}}}$ valeurs pour un i donné. La complexité totale de l'attaque est alors de $(k+1) * \frac{2^\ell}{\binom{\ell}{\frac{d-1}{2}}}$.

Si l'attaquant choisi d'inverser la fonction de hachage, et que la fonction de hachage est bien choisie, alors il n'existe pas d'algorithme meilleur que la recherche exhaustive dont la complexité est de 2^L. Dans le cas où il existerait un algorithme meilleur que la recherche exhaustive, la complexité de l'attaque serait de $(k+1) * \mathcal{C}(\mathcal{H})$.

7 Conclusion

Nous avons présenté dans cet article un schéma d'authentification biométrique ne nécessitant pas de stockage des données biométriques. Notre construction permet de limiter le risque de vol des données biométriques et, de ce fait, améliore la sécurité de l'authentification biométrique. Notre méthode présente en outre l'avantage de pouvoir révoquer la donnée secrète si celle ci était compromise, sans que la donnée biométrique soit compromise pour autant. Notre construction peut s'utiliser en remplacement des applications habituelles de l'authentification biométrique. En outre, la valeur secrète engagée dans le FingerKey offre de nouvelles applications ; par exemple, elle peut être utilisée comme clé cryptographique.

Nous avons proposé une méthode d'authentification biométrique réellement utilisable en pratique, nos résultats expérimentaux ont donné des résultats encourageants. Une voie de recherche pour l'amélioration de notre méthode est d'améliorer la caractérisation de l'empreinte par FingerCode. Nous pensons qu'une méthode hybride utilisant à la fois l'analyse de texture et les minuties comme cela a été introduit dans [JRP01] serait intéressante à étudier.

Références

[CKL03] T. Charles Clancy, Negar Kiyavash, and Dennis J. Lin. Secure smartcardbased fingerprint authentication. In *Workshop on Biometrics Methods and Applications*, pages 45–52, New York, NY, USA, 2003. ACM Press.

[CL06] Ee-Chien Chang and Qiming Li. Hiding secret points amidst chaff. In *EUROCRYPT*, pages 59–72, 2006.

[DRS04] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors : How to generate strong keys from biometrics and other noisy data. In *EUROCRYPT*, pages 523–540, 2004.

[Gab46] D. Gabor. Theory of communication. *Journal of the Institute of Electrical Engineers*, 93 :429–457, 1946.

[JM03] Anil K. Jain and David Maltoni. *Handbook of Fingerprint Recognition*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

[JPHP00] A. Jain, S. Prabhakar, L. Hong, and S. Pankanti. Filterbank-based fingerprint matching. In *Proc. of IEEE Transactions on Image Processing, vol. 9, no. 5, pp. 846–859*, 2000.

[JRP01] A. Jain, A. Ross, and S. Prabhakar. Fingerprint matching using minutiae and texture features. In *Proc. of International Conference on Image Processing (ICIP)*, pages 282–285, Thessaloniki, Greece, October 2001.

[JS02] A. Juels and M. Sudan. A fuzzy vault scheme. In *In Proceedings of IEEE International Symposium on Information Theory, 2002.*, 2002.

[JW99] Ari Juels and Martin Wattenberg. A fuzzy commitment scheme. In *CCS '99 : Proceedings of the 6th ACM conference on Computer and communications security*, pages 28–36, New York, NY, USA, 1999. ACM Press.

[MS77a] F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, 1977.

[MS77b] F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-Correcting Codes, Part II*. North-Holland, 1977.

[Pra01] Salil Prabhakar. *Fingerprint Classification and Matching Using a Filterbank*. PhD thesis, Michigan State University, 2001.

[UJ04] U. Uludag and A.K. Jain. Fuzzy fingerprint vault. In *Proc. Workshop : Biometrics : Challenges Arising from Theory to Practice*, pages 13–16, 2004.

[UPPJ04] U. Uludag, S. Pankanti, S. Prabhakar, and A. Jain. Biometric cryptosystems : Issues and challenges, 2004.