A propos d'un schéma d'authentification OTP

Gilles Dequen, Gaël Le Mahec, Monika Trimoska

A propos d'un schéma d'authentification OTP

Gaël Le Mahec
Laboratoire MIS
Université de Picardie Jules Verne
33 rue Saint-Leu 80000 Amiens
gael.le.mahec@u-picardie.fr

Monika Trimoska
Laboratoire MIS
monika.trimoska@u-picardie.fr

Gilles Dequen
Laboratoire MIS
gilles.dequen@u-picardie.fr

Résumé

L'authentification des utilisateurs d'un système d'information reposant sur la communication d'un couple (Identifiant, Mot de passe) est un processus très répandu. Il présente cependant plusieurs faiblesses qui peuvent s'avérer très dommageables.

La première faiblesse est la nécessité d'une confiance forte de l'utilisateur envers le service auquel il communique ses informations sensibles. Plus précisément, cette relation suppose une confiance dans les intentions dudit service mais aussi dans sa capacité à protéger les données d'identification de l'utilisateur. La seconde faiblesse est sa sensibilité à l'usurpation du service : lorsqu'un attaquant est en mesure de se faire passer pour le service attendu (campagne de phishing, honeypots, ...) il obtient très facilement les données d'identification de l'utilisateur.

Ces faiblesses sont d'autant plus gênantes que comme l'attestent plusieurs études, les utilisateurs de services en ligne sont une majorité à utiliser le même mot de passe pour la plupart des services auxquels ils se connectent. Ainsi, l'obtention d'un mot de passe pour un service particulier donne bien souvent accès à la quasi-totalité des services en ligne pour lesquels les utilisateurs disposent d'une autorisation d'accès.

Nous proposons ici un protocole d'identification permettant de résoudre ces problèmes tout en conservant le système identifiant/mot de passe bien connu des utilisateurs.

Mots clefs

Authentification, Keccak, OTP

1 Introduction

L'authentification est la capacité pour tout un chacun de pouvoir garantir son identité et s'assurer de celle de son interlocuteur. Au delà de la sensibilité des échanges futurs, c'est une question cruciale fondant l'accord de confiance. Ainsi, aussi bien dans le cadre professionnel (lecture de mails, applications « métier », etc.) que privé (réseaux sociaux, achats en ligne, etc.), nous sommes quotidiennement amenés à faire confiance aux procédures de sécurisation d'accès aux systèmes d'information (SI) que nous utilisons. Même si les

solutions d'authentification basées sur la biométrie sont de plus en plus répandues, la communication d'un couple (Identifiant, Mot de passe) reste majoritaire.

A l'instar des solutions basées sur la biométrie [1], l'usage du couple (Identifiant, Mot de passe) présente plusieurs faiblesses qui peuvent s'avérer dommageables aux utilisateurs. La première faiblesse est la nécessité d'une confiance forte de l'utilisateur envers le service auquel il transmet son mot de passe, confiance dans ses intentions mais aussi dans sa capacité à protéger ses données d'identification. La seconde faiblesse est sa sensibilité à l'usurpation du service. Dans ce cas de figure, l'attaquant est en mesure d'obtenir les données d'identification de l'utilisateur. Ces faiblesses sont d'autant plus gênantes que des études ont montré [2] que les utilisateurs de services en ligne sont une majorité à utiliser le même mot de passe pour la plupart des services auxquels ils se connectent fragilisant d'autant plus leur ecosystème de services en ligne.

Le présent article présente une solution d'authentification forte, nommée `CrypTonAuth`, basée sur l'usage original de la fonction `Keccak` fondant le standard de hachage cryptographique `SHA3` [3]. CrypTonAuth utilise pour cela la technologie CrypTonID qui permet la reconstruction de données hachées à partir d'une information complémentaire en utilisant le principe de l'attaque « midgame » [4].

In fine, `CrypTonAuth` constitue une solution d'authentification du type « One-Time-Password » souple pour l'utilisateur car n'exigeant aucun changement de procédure en maintenant la séquence de saisie (Identifiant, Mot de passe).

CrypTonID et CrypTonAuth font l'objet de deux brevets distincts déposés par la SATT Nord pour le compte du laboratoire MIS dont l'un a déjà été acquis par la société Dhimyotis spécialisée dans la sécurité informatique et la « confiance numérique ».

La suite de cet article se divise en trois parties où nous rappelons dans un premier temps les principes (cf. 2), les risques éventuels (cf. 2.1) ainsi que les possibles contre-mesures (cf. 2.2) à l'usage du couple (Identifiant, Mot de passe) pour l'authentification. Dans un second temps, nous décrivons brièvement les principes de la technologie `CrypTonID`. La dernière partie explicative (cf. 4) est consacrée à la présentation du protocole `CrypTonAuth` et propose un récapitulatif de ses avantages comparativement aux principales méthodes couramment utilisées.

2 L'authentification par login/mot de passe

L'authentification des utilisateurs par mot de passe est probablement la procédure la plus couramment employée sur les systèmes d'information. Cependant, ses limites et les risques qu'elle induit ont conduit à s'interroger sur des méthodes de remplacement. Des méthodes très sûres basées sur la cryptographie asymétrique jusqu'aux méthodes biométriques dont la sûreté est régulièrement remise en cause, aucune ne semble à ce jour prendre le pas sur ce procédé bien connu des utilisateurs. Cependant, avec la multiplication des services en ligne à usage personnel comme professionnel, les risques liés à ce mode d'authentification deviennent un point critique de la sécurité des systèmes d'information.

2.1 Risques

2.1.1 Multiplication des mots de passe et contournements

Avec la multiplication des services en ligne, un utilisateur « moyen » pourrait potentiellement avoir à retenir des dizaines de mots de passe dont les bonnes pratiques de sécurité voudraient qu'ils soient tous différents (Réseaux sociaux, forums, sites marchands, sites d'information, plateforme de musique en ligne, sites bancaires, boîte mail,...) et régulièrement renouvelés. Cela tend à favoriser les contournements du « bon usage » des principes de sécurité élémentaires. Dans le meilleur des cas, les usagers utilisent un gestionnaire de mots de passe, version informatisée et parfois a priori plus sécurisée du « post-it sous le clavier ». Dans le pire

des cas, ils utilisent un mot de passe facile à retenir, bien souvent « faible » et dont la cryptanalyse est à une portée acceptable quand elle n'est pas triviale (l'emploi de «12345», «azerty» etc, reste malheureusement très fréquent). Le vol du mot de passe devient alors une porte d'entrée universelle vers des pans entiers de la vie numérique des personnes visées avec tous les risques que cela induit. Pour finir, il est assez rare que les usagers maintiennent à jour une liste exhaustive des services auprès desquels ils sont enregistrés induisant de fait une vulnérabilité durable y compris lorsque le vol du mot de passe est découvert.

2.1.2 Confiance dans les services et leur capacité à protéger les données

L'utilisation d'un mot de passe pour s'authentifier sur un service en ligne implique que l'utilisateur fasse a minima confiance, au fournisseur du service quant aux usages de cette information sensible. Cette confiance n'est pas suffisante. Un fournisseur de service, y compris bien intentionné, doit être en mesure de protéger efficacement les séquences d'identification de ses usagers. Or, compte tenu de la multiplicité des services, de leurs pratiques et des moyens à disposition de ceux-ci, il est très difficile pour eux d'offrir de telles garanties.

2.1.3 Phishing, usurpation d'identité, malwares

Une façon très simple et qui semble plutôt efficace pour obtenir les identifiants et mots de passe des utilisateurs est tout simplement de les leur demander. C'est sur ce principe que repose les campagnes de hameçonnage (phishing) par mail ou conduisant les utilisateurs à entrer leurs identifiants sur un site contrefait d'un service en ligne. L'installation involontaire d'un malware est un autre vecteur de fuite des mots de passe au sein d'un SI.

2.2 Contre-mesures

2.2.1 Authentification par cryptographie asymétrique

Le meilleur moyen de ne pas se faire voler son mot de passe est de ne pas en avoir : la cryptographie asymétrique permet de remplacer l'exposition d'un mot de passe par un challenge cryptographique consistant pour l'utilisateur à démontrer qu'il dispose d'une information secrète (la clé privée) mathématiquement liée à une information publique (la clé publique) connue de tout le monde et par conséquent du serveur. Répandu pour les connexions type `ssh` ou VPN, l'authentification d'un client peut-être également réalisée pour de nombreux autres services, comme par exemple les connexions `https`. Très efficace, ce type d'authentification reste cependant assez complexe à mettre en œuvre suivant le service concerné. Par ailleurs, les méthodes de chiffrements asymétriques « classiques » nécessitent des ressources de calcul (et donc énergétiques) conséquentes, actuellement hors de portée de petits objets alimentés par pile ou batterie sur des longues durées.

2.2.2 Protection des mots de passe enregistrés

Pour éviter qu'une intrusion sur un SI révèle les mots de passe des utilisateurs, ceux-ci sont en général chiffrés ou hachés avant d'être enregistrés. De plus, afin de contrer les éventuelles attaques à textes clair, une valeur aléatoire (le « grain de sel ») est ajoutée au mot de passe avant chiffrement. Ce « grain de sel » est alors enregistré avec l'image chiffrée du mot de passe. Dans ce système, les mots de passe « faibles » (trop simples, trop fréquemment utilisés, ...) résistent peu aux attaques par dictionnaire ou par compromis temps-mémoire (par exemple par l'utilisation de tables « Arc-en-Ciel »).

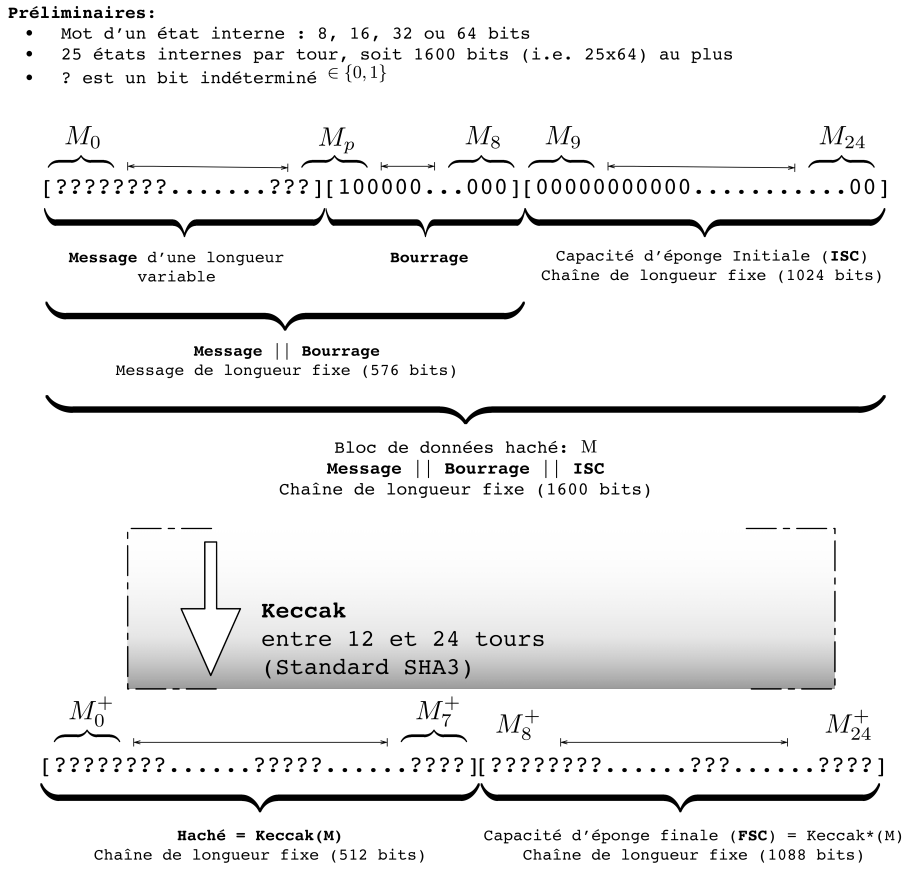

Figure 1 - Résumé du processus de hachage `Keccak` *sur un bloc de données*

3 CrypTonID

Les fonctions de hachage cryptographiques sont des fonctions qui prennent en entrée une donnée de taille quelconque et retournent une image de taille fixe (image au sens mathématique de l'image par une fonction). Elles ont pour propriété :

— d'être résistante à la première préimage : quelle que soit l'image (fixée), il est très difficile de trouver une donnée (quelconque) qui donne cette image

— d'être résistante à la seconde préimage : quelle que soit la donnée (fixée), il est très difficile de trouver une donnée différente (quelconque) telle que leurs deux images sont identiques

— d'être résistante aux collisions : il est trés difficile de trouver deux données (quelconques) dont les images sont identiques

La fonction `Keccak` constitue le fondement du standard cryptographique `SHA3`, sélectionné par le NIST en octobre 2012 et standardisé en 2015 [3]. Cette fonction de hachage cryptographique est par ailleurs le socle de `CrypTonID` [5]. La figure 1 propose d'illustrer de façon synthétique les différents éléments de hachage `Keccak` pour un unique bloc de données en entrée se limitant au maximum à 72 octets (soit 576 bits) permettant de calculer un haché de 512 bits (cf [3], p.20). Ainsi, pour une capacité globale de 1600 bits, les capacités d'éponges initiale (notée `ISC`) et finale (notée `FSC`) sont respectivement de 1024 bits et de 1088 bits.

La technologie `CrypTonID` est basée sur le principe de l'attaque dite « midgame » permettant de reconstruire une donnée, à partir d'un modèle algébrique de la fonction `Keccak` d'un haché et d'une informa-

tion complémentaire obtenue à la fin du hachage (la capacité d'éponge finale FSC- voir [6]). Le modèle algébrique utilisé est une expression logique (propositionnelle) et le moteur de résolution est un solveur sat spécifiquement mis au point pour cette tâche. Au final, cette approche s'apparente à un « puzzle cryptographique » dans la mesure où la validité de l'assemblage entre le haché et le FSC (nommé *BackHash* dans le cadre de CrypTonAuth) est vérifiable au terme de la reconstruction. Cette propriété est celle qui est utilisée au sein du protocole d'authentification CrypTonAuth.

4 CrypTonAuth

En s'appuyant sur les fonctionnalités de CrypTonID [5], on peut définir un protocole d'identification, toujours sur le modèle *login/password*, mais supprimant une bonne part des risques qu'il induit potentiellement (cf section 2.1). Le protocole assure notamment qu'aucune information sensible ne transite par le réseau. Ainsi, même en cas d'interception des échanges ou en cas de corruption du serveur, un attaquant ne pourra pas obtenir d'information sur le mot de passe utilisé. Par ailleurs, le protocole utilisant une information supplémentaire, sur le modèle de l'identification à deux facteurs, le vol ou la découverte du mot de passe n'est pas suffisant pour usurper l'identité d'un utilisateur. Dans le pire des cas, lorsque le second facteur d'identification est lui-même corrompu, l'utilisateur est alerté dés sa prochaine tentative de connexion.

Une des caractéristiques du protocole est sa dissymétrie en terme d'utilisation de ressources entre les deux parties impliquées dans l'identification : essentiellement le calcul de *hachés* en utilisant Keccak [3]- spécifiquement conçu pour nécessiter très peu de ressources en calcul et en mémoire - ou au contraire la résolution d'un problème de satisfiabilité booléenne (un problème sat) grâce à un solveur spécifique nécessitant beaucoup plus de ressources. Les deux opérations de CrypTonID étant parfaitement symétriques, le protocole permet de choisir qui du client cherchant à s'identifier ou du serveur aura les tâches de calcul les plus « lourdes » à réaliser. Par exemple, lorsque le protocole est utilisé pour se connecter à un ordinateur à l'aide d'un petit objet à faibles capacités, c'est le serveur qui accomplira la reconstruction. À l'inverse, pour ouvrir une porte à l'aide d'un smartphone, la serrure (serveur) étant présumée limitée en ressource, c'est le client (smartphone) qui réalisera cette tâche. Dans la suite de l'article, pour en faciliter sa compréhension, c'est le serveur qui effectue les tâches coûteuses de reconstruction.

Le protocole prévoit plusieurs modes de fonctionnement permettant de l'adapter au mieux à son usage. On pourra ainsi décider d'utiliser : un stockage local pour conserver le second facteur d'identification ; un serveur externe chargé de le transmettre ; une application pour smartphone ; une carte à puce ; un objet communiquant à courte portée (NFC, BlueTooth, ...) ; ...

Cette souplesse dans la configuration du protocole permet enfin d'introduire un mode de fonctionnement original qui permet d'identifier un utilisateur en combinant, via CrypTonID, le partage du second facteur et du mot de passe en fonction du niveau de sécurité souhaité. Un utilisateur pourra par exemple décider que lorsqu'il est en possession de son smartphone, il est inutile de lui demander un mot de passe pour se connecter sur son ordinateur personnel ; qu'à l'inverse, il lui faudra à la fois son smartphone, un objet communiquant dédié et un mot de passe pour se connecter au serveur d'applications de son lieu de travail. On peut ainsi imaginer des niveaux de sécurité différents en fonction du contexte, configurables par l'utilisateur.

La section suivante explicite le fonctionnement du protocole et des variantes principales utilisables pour l'identification des utilisateurs.

4.1 Fonctionnement du protocole

La première phase du protocole permet l'enregistrement d'un nouvel utilisateur. Il s'agit d'une phase critique durant laquelle les deux parties échangent une information secrète qui permettra d'authentifier le serveur et d'assurer l'unicité de chaque connexion en empêchant le rejeu pour les connexions ultérieures. Cette phase doit donc être réalisée via une connexion sécurisée. Il peut s'agir d'une connexion sécurisée par TLS, d'une connexion physique directe du périphérique, etc. Après cette première phase, le client et le serveur partagent une information secrète commune (256 bits générés aléatoirement).

À ce stade, le dispositif utilisateur demande d'entrer un mot de passe ou en génère un nouveau aléatoirement (sur 256 bits également). Un *haché* `CrypTonID`, calculé à partir de la valeur de contrôle et du mot de passe est transmis au service.

À la suite de cette première étape d'enregistrement, pour chaque identification, le protocole suit le schéma de la figure 2

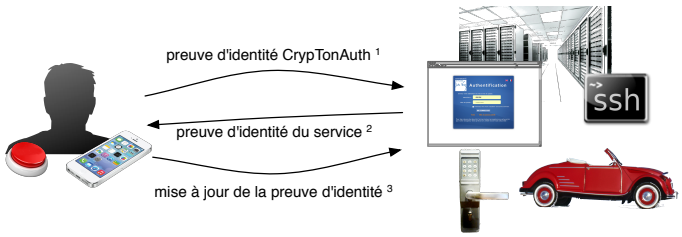

Figure 2 - Schéma général du protocole d'identification `CrypTonAuth`

1. L'utilisateur transmet sa preuve d'identité `CrypTonID` permettant au service de reconstruire une valeur de contrôle connue du service et du client.
2. Le serveur calcule ou obtient d'un service extérieur une nouvelle valeur de contrôle. Il transmet le *haché* de cette valeur comme preuve de son identité. De son côté, le client calcule le haché à partir de la valeur qu'il a lui-même obtenue ou calculée. Si les hachés sont identiques, le client a l'assurance de l'identité du service.
3. Le client recalcule un *haché* `CrypTonID` à partir de la valeur de contrôle et d'un mot de passe (le mot de passe peut être différent du mot de passe précédent). Le *haché* est transmis au service pour une identification ultérieure.

Lors de l'étape 1, la donnée transmise correspond au *BackHash* qui, associée au *haché* de la valeur de contrôle, permet de reconstruire cette valeur. Pour calculer ce *BackHash*, il est nécessaire de connaître le mot de passe ayant servi à la création du *haché*. Ce calcul s'effectue en deux étapes, garantissant qu'il est aussi difficile de retrouver le mot de passe que d'inverser la fonction `Keccak`. À cette étape, le mot de passe peut-être directement entré par l'utilisateur ou bien lui-même reconstruit à partir de plusieurs opérations `CrypTonID` effectuées à partir d'autres périphériques entrant dans le processus d'identification `CrypTonAuth`. La figure 3 illustre ce fonctionnement.

Dans l'étape de répartition, chaque entité effectue une opération `CrypTonID` pour décomposer le mot de passe et le secret partagé. Le *BackHash* est transmis à l'entité suivante dans la chaîne de répartition puis effacé. Pour reconstruire les données d'identification, chaque entité reçoit le *BackHash* de l'entité précédente dans la chaîne de reconstruction. À la fin de la chaîne, la donnée initiale est reconstruite.

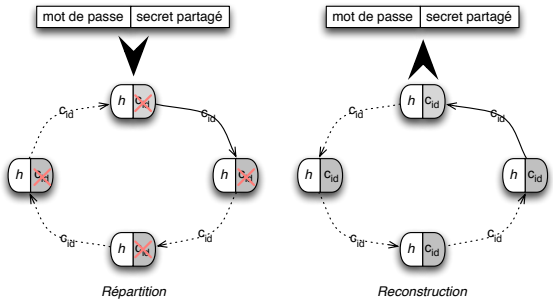

Figure 3 - Processus de répartition et reconstruction des données d'identification `CrypTonID`

À l'étape 2 de l'identification, la valeur de contrôle (256 bits aléatoires) peut-être obtenue algorithmiquement comme le résultat d'une opération de hachage `Keccak` dépendant du secret partagé et de la donnée d'identification. Cette donnée dépendant elle-même d'une valeur aléatoire et d'un mot de passe connu seulement de l'utilisateur, elle a dans ce cas, une qualité d'aléatoire cryptographique. On peut également choisir de déléguer la génération de cette valeur de contrôle à un service ou un périphérique externe. Dans ce cas, le client devra être en mesure d'obtenir cette même valeur de son côté. Elle peut par exemple être transmise par SMS sur le modèle des identifications à deux facteurs grand public *classiques*, via une application pour smartphone, par l'intermédiaire d'un site tiers de confiance, etc. Le service transmet alors le haché de cette valeur permettant au client de s'assurer de son identité.

À la dernière étape, le client transmet la nouvelle donnée d'identification calculée à partir de la valeur de contrôle et d'un mot de passe qui peut être changé arbitrairement par le client sans que le service puisse le détecter. C'est cette donnée qui sera utilisée à la connexion suivante.

4.2 Avantages

4.2.1 Coût algorithmique/énergétique

`CrypTonID` reposant sur la fonction de hachage `Keccak`, le calcul des éléments d'identification de `CrypTonAuth` côté client est particulièrement rapide et peu coûteux en énergie. `Keccak` est en effet construit sur des opérations binaires (XOR, ROT, AND, NOT) généralement réalisées en un seul cycle processeur pour un coût énergétique très faible. Nous avons facilement implémentés toute la phase cliente pour une transmission des données via BlueTooth sur un micro-contrôleur 8 bits ne disposant que de 32Ko de mémoire partagée par le code et les données.

4.2.2 Forte adaptabilité

Comme nous l'avons vu dans les sections précédentes, les différentes étapes de `CrypTonAuth` offrent de nombreuses possibilités de paramètrages et d'implémentations. `CrypTonAuth` permet ainsi de choisir facilement un modèle d'identification en fonction du contexte et du niveau de sécurité choisi.

4.2.3 Vulnérabilité réduite aux mots de passe faibles

Par nature, `CrypTonAuth` utilise un second facteur d'identification qui réduit très largement les risques d'utilisation d'un mot de passe faible. Le mot de passe peut en effet s'apparenter à un code PIN ou une *passphrase* permettant de générer une preuve d'identité.

4.2.4 Récapitulatif

	CrypTonAuth	Autres méthodes d'identification
Complexité & Coût énergétique	Très faible côté client, plus conséquent côté serveur pour une authentification forte non vulnérable à l'interception des communications	Pour un même niveau de sécurité, nécessite l'emploi de cryptographie asymétrique côté client et serveur
Détection d'une usurpation à la reconnexion	La détection est incluse dans le protocole	En général, une procédure de détection doit être spécifiquement implémentée
Mise à jour du mot de passe	Totalement transparent pour le serveur	Une procédure spécifique doit être mise en place côté serveur
Sensibilité aux mots de passe « faibles »	L'utilisation d'un mot de passe faible n'altère pas la sécurité	Seuls les protocoles utilisant la cryptographie asymétrique permettent l'utilisation d'un mot de passe faible sans risque
Authentification du serveur	Incluse dans le protocole et non utilisable par un tiers	Avec des certificats numériques traçables par tous

5 Conclusion

À ce jour CrypTonAuth a été implémenté pour l'authentification d'un utilisateur sur un système Unix via un module PAM et pour la connexion à un site web. Les connexions s'effectuent en utilisant un objet communiquant en BlueTooth Low Energy sans appariement préalable. Nous pensons qu'il représente un moyen sûr et pratique d'authentifier des utilisateurs avec un haut niveau de sécurité, adaptable à des contraintes sécuritaires diverses. Sa simplicité d'utilisation devrait par ailleurs permettre une bonne acceptabilité contrairement à des alternatives très contraignantes qui conduisent parfois à des stratégies de contournement de la part des utilisateurs.

Bibliographie

[1] A. Hadid. Face biometrics under spoofing attacks : Vulnerabilities, countermeasures, open issues, and research directions. Dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 113–118, 2014.

[2] A. Das, J. Bonneau, M. Caesar, N. Borisov et X. Wang. The tangled web of password reuse. Dans *NDSS*, volume 14, pages 23–26, 2014.

[3] N. F. Pub. Fips pub 202. sha-3 standard : Permutation-based hash and extendable-output functions. Dans *Federal Information Processing Standards Publication*, 2015.

[4] Donghoon C. et Moti Y.. Midgame attacks. Dans *CRYPTO, rump session*, 2012.

[5] Dequen G., Legendre F. et Krajecki M.. Hash data retrieval method. *Patent N. 15305475.4*, page 38p., 2015.

[6] G. Bertoni, J. Daemen, M. Peeters et G. Van Assche. The keccak reference. Submission to NIST (Round 3), 2011.