Principes de fonctionnement des signatures électroniques

En vue de la préparation à l'usage des communications signées et chiffrées sur Internet

---

Patrick Legand
Consultant SSI
contactPL@patrick-legand.com
http://blog.patrick-legand.com

- Septembre 2006 -

Notion de signature électronique

À l'heure de la dématérialisation des procédures administratives et du développement de l'échange par voie électronique, il est capital de disposer d'un moyen permettant d'établir les conditions de la confiance et, notamment, de garantir l'authenticité d'un document. Ce moyen se nomme *Signature électronique*.

La signature électronique est un élément fondamental sans lequel la notion d'acte officiel sur Internet n'existerait pas. Au même titre que la signature manuscrite, la signature électronique fournit la preuve de l'authentification de l'origine d'un document, de l'intégrité d'un document, de la validité d'informations importantes comme l'horodatage, et rend impossible la répudiation d'un acte.

Le site de la DCSSI (www.dcssi.gouv.fr) traite de façon très complète la problématique de la signature électronique, des aspects politiques, juridiques et organisationnels qui s'y rapportent.

Dans cet article, nous nous limitons uniquement à présenter les principes techniques conduisant à l'élaboration d'une signature électronique, dans le but de faciliter la compréhension des mécanismes de gestion des certificats, de sécurisation de la messagerie et des transactions électroniques.

Sur le plan technique, une signature électronique repose sur deux opérations mathématiques: l'élaboration d'un condensé cryptologique (aussi appelé empreinte numérique ou code de hachage), et une opération de chiffrement à l'aide d'un algorithme à clés publiques, tel que RSA (se reporter au document traitant des principes de fonctionnement de RSA sur le même site).

Fonctions de hachage

Une fonction de hachage est une fonction mathématique dont nous n'allons pas détailler ici les mécanismes: ils n'aident pas à la comprehension du service rendu et sortent largement du cadre de cet article. Le lecteur désireux d'approfondir ses connaissances en cryptologie pourra se référer à l'excellent ouvrage de Bruce Schneier « Cryptographie Appliquée », publié aux éditions International Thomson Publishing.

En revanche, il est important de retenir ceci: une fonction de hachage a pour objet de convertir une chaîne d'une longueur arbitraire (un fichier texte, un fichier exécutable de plusieurs méga-octets, un fichier musical....) en une chaîne de caractères de taille fixe, qui s'étend le plus souvent sur 128, 160, 256, voire 512 bits.

Cette chaîne de caractères, ou empreinte numérique, possède des propriétés tout à fait remarquables:

- Sur Internet, on part du principe qu'une empreinte numérique correspond à un document unique (cette affirmation n'est pas rigoureusement exacte. Voir l'encadré ci-dessous).
- Une modification à l'intérieur d'un document, même infinitésimale, provoque un changement radical de son empreinte: toute modification devient ainsi immédiatement détectable.
- Les fonctions de hachage sont des fonctions à sens unique: il est aisé de calculer l'empreinte numérique d'un document, mais il est très difficile de retrouver le document initial à partir de son empreinte. Lorsqu'une empreinte numérique est publiée, on part encore du principe qu'aucune information relative au contenu du document associé n'est divulguée.

Les principaux algorithmes de hachage utilisés actuellement sont:

- MD5 (MD pour Message Digest): MD5 crée une empreinte de 128 bits. MD5 était un très bon algorithme, mais l'augmentation de la puissance de calcul des ordinateurs et la progession des techniques de cryptanalyse le rendent aujourd'hui moins sûr.

– SHA-1 (Secure Hash Algorithm): SHA-1 crée des empreintes de 160 bits. Il est considéré plus fiable que MD5.

– SHA-256 / SHA-512, beaucoup plus sûrs mais moins répandus, créent respectivement des empreintes de 256 et 512 bits.

Attaque des anniversaires

L'hypothèse selon laquelle une empreinte numérique correspond à un document unique n'est pas tout à fait vraie. Même si l'ensemble formé par les nombres de 512 bits représente un espace colossal (voir à ce sujet le document consacré à RSA), cet espace est fini. Il en résulte fatalement que deux, voire plusieurs messages, puissent très bien correspondre à une seule et même empreinte numérique. Ce phénomène s'appelle une « **collision** ».

Bien entendu, lorsque l'algorithme cryptologique est fiable et, surtout, lorsqu'il est bien utilisé, la probabilité d'une collision est extrêmement faible. Cependant, en mettant en oeuvre une attaque dite « des anniversaires », cette probabilité augmente dans des proportions vertigineuses.

Une attaque classique contre les fonctions de hachage s'appuie sur ce que l'on appelle le **« paradoxe des anniversaires »**:: combien faut-il réunir de personnes dans une assemblée pour avoir une bonne chance de rencontrer une personne née le même jour que vous? Une année comportant 365 jours (faisons abstraction des années bisextiles), l'on pourrait s'attendre à un nombre conséquent. Effectivement, la réponse est 253. Maintenant examinons ceci: combien de personnes doivent être réunies dans une pièce pour que deux d'entre elles aient de bonnes chances d'être nées le même jour? Cette fois la réponse est tout à fait différente: seulement 23 ! Dans le premier cas, vous fixez deux conditions: votre date d'anniversaire, et le fait que deux personnes soient nées le même jour. Dans le second cas, une seule condition est imposée: le fait que deux personnes soient nées le même jour; peu importe la date.

Cet exemple illustre bien la fameuse attaque des anniversaires dirigée à l'encontre des fonctions de hachage. Imaginons le protocole suivant:

– Alice souhaite établir un contrat avec Bernard. Elle prépare donc un document que Bernard jugera tout à fait honnête,

– En secret, Alice rédige une autre version de ce contrat, cette fois beaucoup plus favorable à Alice,

– Alice apporte au contrat une modification mineure et indécelable (par exemple l'insertion d'un espace à la fin d'une ligne), calcule l'empreinte numérique du document et l'enregistre dans une table. Elle réitère ce même processus des dizaines de millions de fois, en prenant bien soin d'enregistrer chaque nouvelle empreinte dans la table.

– Alice fait de même avec le deuxième document.

– Elle compare ensuite les deux tables à la recherche de deux valeurs identiques.

S'il existe effectivement une collision, Alice pourra prétendre que Bernard a signé le faux contrat et, par la suite, l'attaquer en justice.

Plus généralement, si l'empeinte numérique comporte *n* bits, on démontre mathématiquement qu'en calculant $2^{n/2}$ empreintes numériques, la recherche de collisions a plus d'une chance sur deux d'aboutir.

Rappel: un tel stratagème peut réellement fonctionner uniquement si le code de hachage du document initial n'est pas fixé à l'avance. Pour éviter de se faire piéger:

– Fixer soi-même cette valeur: générer soi-même le document ou, au besoin, imposer une modification mineure sur la version finalisée du document soumis.

– Veiller à ce que l'empreinte numérique soit élaborée avec un bon alogorithme de hachage (ex. SHA). Un un tel algorithme offre des propriétés de **résistance à la collision** (il est difficile de trouver deux messages correspondant à la même empreinte numérique).

A titre indicatif, on rencontre dans la littérature de nombreuses dénominations pour désigner la valeur produite par un algorithme de hachage: empreinte numérique, condensé cryptologique, checksum cryptologique, résumé, code de hachage, valeur de hachage, digest, etc. Toutes ces dénominations veulent sensiblement dire la même chose.

Exemples d'utilisation

Examinons de plus près l'utilisation des empreintes numériques en se basant sur des exemples concrets.

Calculer une empreinte numérique est extrêment simple. Commencez d'abord par créer un petit fichier texte (vous pouvez notamment vous servir du Bloc-notes):

> « C'est un plaisir de faire sauter l'ingénieur avec son propre pétard.
> HAMLET »

Sauvegardez ce fichier sous un nom quelconque, par exemple « essaiEmpreinte.txt » et calculez son condensé cryptologique SHA-1. Vous réalisez très facilement cette opération à l'aide d'un petit utilitaire « **sha1sum.exe** », téléchargeable gratuitement sur Internet. Vous obtenez la valeur suivante:

```
BFCD:5B5B:E4E2:FC43:D56A:6B56:7031:A316:DBE5:5991
```

De même, amusez vous à calculer l'empreinte SHA-1 d'un fichier plus volumineux, comme celui d'une vidéo. Vous constatez que l'on obtient toujours une valeur de 160 bits.

Cette notion de « résumé », d'« empreinte » ou de « condensé » prend ici tout son sens: une telle valeur ne dit absolument rien à propos du contenu du fichier, mais elle indique en un clin d'oeil si celui-ci a été modifié par rapport à une version précédente. Essayez de changer très légèrement le contenu du fichier texte créé il y a quelques instants, en supprimant par exemple uniquement le point à la fin de la phrase. Calculez à nouveau le checksum cryptologique du fichier:

```
032B:B5A2:16C9:15DC:CF07:9E2A:11A4:27A0:48ED:057F
```

Constatez l'énorme différence entre les deux valeurs obtenues! Bien sûr, un oeil perspicace aurait peut-être détecté l'absence du point dans le fichier modifié. Mais imaginez l'ajout d'une phrase sibylline à l'intérieur d'un contrat de trois cents pages!

Dans le cadre de l'échange de documents électroniques sur Internet, une empreinte numérique revêt donc de multiples intérêts. Supposons par exemple que vous souhaitiez envoyer un message chiffré à un correspondant; vous lui demandez sa clé publique, il vous l'envoie par e-mail, ou, plus simplement, vous la récupérez à partir d'un annuaire. Si l'on s'appuie sur une implémentation (très) artisanale de RSA disponible sur ce même site, supposons que la valeur de cette clé publique soit: e = 771, n = 15853. Dans ce cas précis, il est facile de vérifier via une autre source (le téléphone, un autre annuaire, le site web de votre correspondant...) que ces valeurs constituent effectivement la clé publique de votre interlocuteur.

Qu'en est-il dans la réalité, lorsque ces nombres atteignent des valeurs astronomiques? Voici l'exemple d'une « vraie » clé publique RSA:

```
« 30818902818100a32d946ea519646f84109e62548bfa7050c5ab378
fbfd4ac099815c1edb2e4530f1de18033ea79c4f371f0784f24828668
4220b240a9fc2fc17879d28bbe916518292db70b62c5e0aaea56fa32a
534ec5162706b2b63d79d222167c549e0546a0194384c685e56619169
0ff7baa1f867d5983c7b78b573dcee4b51a29f72c0070203010001 »
```

Allez vous passer du temps à téléphoner à votre correspondant pour vérifier bit à bit que cette valeur est exacte? Evidemment non! Le plus simple consiste alors pour le correspondant à publier dans un annuaire l'empreinte numérique de sa clé:

```
Empreinte MD5:
A031:51E1:D1CD:DE6D:3C06:5185:C6FA:F80A

Empreinte SHA-1:
FE50:98DD:C8EC:2E6B:EE74:77CF:DDAC:0196:FD3B:D8DD
```

De votre côté, vous téléchargez cette clé, puis recalculez son empreinte numérique à l'aide soit de md5sum.exe, soit de sha1sum.exe; vous effectuez ensuite la comparaison. En quelques secondes, vous saurez si vous avez téléchargé la bonne clé.

De même, vous téléchargez un logiciel. Comment savoir si vous rapatriez le bon logiciel, ou une version piratée? Vous n'allez certainement pas inspecter le code avant de l'utiliser. Un

moyen simple consiste encore à s'appuyer sur les empreintes numériques. Aujourd'hui, tous les éditeurs – ou presque ! – publient sur leurs sites au minimum les condensés MD5 et SHA-1 de leurs logiciels; à vous de faire le nécessaire pour vérifier si la version chargée sur votre poste est bien celle de l'éditeur.

Certes, de telles informations ne sont pas passionnantes à manipuler, mais elles représentent un moyen simple et efficace pour éviter de manipuler bon nombre d'objets corrompus!

Notez au passage qu'aucun secret n'est mis en oeuvre au cours de l'opération de hachage. Les algorithmes de hachage sont publics, aucune clef n'est requise. Les codes de hachages sont publics, tout le monde peut calculer l'empreinte numérique d'un fichier, tout le monde peut vérifier une empreinte et détecter une éventuelle compromission.

Signature électronique

Entrons maintenant dans le vif du sujet.

Pour simplier, raisonnons sur un exemple concret. Supposons qu'Alice lance un ordre d'achat sur Internet (cet ordre d'achat est matérialisé ici par un petit fichier texte édité à l'aide du Bloc-notes):

« Achetez les 6 Vermeer à n'importe quel prix »

Toutefois, avant d'adresser ce message à Bernard, l'acheteur qui va réaliser l'opération, Alice veut être sûre que cet ordre sera bien compris. Elle effectue alors quelques opérations préliminaires:

- elle calcule l'empeinte numérique SHA-1 de ce fichier,
- chiffre cette valeur ***avec sa clé privée***,
- prépare ensuite un message constitué du fichier texte initial, de la valeur chiffrée qu'elle vient de calculer et de sa clé publique:

  « Achetez les 6 Vermeer à n'importe quel prix »

  « 1949 38750 41029 11039 26928 22558 45045 25432 46635 15171 43797 34568 30566 42513 33950 8337 45045 30182 18020 40340 8822 44994 40340 30705 19988 33950 27196 33368 18992 45045 39714 34958 21506 »

  « Clé publique Alice: e = 15709, n = 48689 ».
- Elle envoie le tout à Bernard.

Dès à présent, notons que, seule, Alice a pu calculer cette valeur chiffrée. Personne d'autre n'a été en mesure de le faire car, seule, Alice est en possession de sa clé privée.

Bernard reçoit donc le message et déchiffre la succession de nombres à l'aide de RSA et de la clé publique d'Alice: 1949^{15709} mod 48689, 38750^{15709} mod 48689, etc. Bernard, qui dispose du même utilitaire artisanal employé par Alice dans cet exemple (encore une fois attention, il s'agit d'un utilitaire développé à des fins pédagogiques et qui n'a rien de standard!), finit par retrouver le texte clair qu'Alice avait chiffré (le lecteur courageux saura aussi retrouver cette valeur):

« F966:D09E:AB27:A868:447E:BACB:1258:EBE4:CD1D:C701 »

Il s'agit de toute évidence d'un code de hachage SHA-1, vraisemblablement l'empreinte numérique du texte du message. À l'aide de sha1sum.exe, Bernard recalcule donc l'empreinte numérique du fichier texte reçu et trouve la valeur suivante:

« F966:D09E:AB27:A868:447E:BACB:1258:EBE4:CD1D:C701 »

La comparaison est évidente: les deux valeurs sont identiques.

Nous pouvons donc déduire de ce résultat plusieurs conclusions:

- le fichier texte émis par Alice n'a subi aucune modification au cours de son transfert sur le réseau, **il est intègre**. Si un pirate s'était avisé à modifier le contenu du message, par exemple « Achetez les **2** Vermeer à n'importe quel prix », Bernard l'aurait détecté en recalculant l'empreinte SHA-1 de ce message
- Bernard a été capable de déchiffrer une empreinte correcte avec la clé publique d'Alice. Cela **PROUVE** qu'Alice a obligatoirement chiffré l'empreinte du message, et personne d'autre: **Alice est donc authentifiée avec certitude en tant qu'émetteur de ce message**.
- Alice ne pourra plus nier avoir émis ce message. Il y **non répudiation** de l'acte.

Cette valeur chiffrée est ce que l'on appelle une *signature électronique*, elle garantit l'authenticité de l'origine et l'intégrité du message.

Une signature électronique n'est autre que le chiffrement de l'empreinte numérique d'un document avec la clé privé de l'émetteur. Elle garantit:

- **l'inégrité** du document signé,
- **l'authentification** de l'émetteur du document,
- **la non répudiation** de l'acte.

Pour plus de renseignements sur la signature électronique, consulter le site de la DCSSI.