Certificats X.509
Quelle confiance leur accorder ?

Giles Carré
RSSI / Centre des Services Numériques / INSA de Toulouse
135 avenue de Rangueil
31077 Toulouse Cedex 4

Résumé

La sécurité des systèmes d'information repose fortement sur le chiffrement asymétrique à clé publique / clé privée et sur la carte d'identité numérique qui l'accompagne, le certificat. Ce dernier permet d'associer une clé publique de chiffrement à une identité (personne physique ou morale) et de certifier cette association.

Le RGS, puis la PSSIE, consacrent l'usage des certificats, en insistant sur la notion de confiance. Si les concepts et les usages des certificats sont bien définis, leur déploiement est assorti de nombreuses difficultés susceptibles d'atteindre leur crédibilité.

L'article rappelle tout d'abord quelques principes liés aux certificats (chiffrement asymétrique, constitution, destination, autorités et politique de certification, etc). En seconde partie, il aborde un certain nombre de problèmes dont la prise en compte est du ressort des systèmes d'information des organismes, et propose quelques bonnes pratiques pour des cas élémentaires.

Mots-clefs

Certificats, X.509, IGC, PKI, autorité de certification, confiance.

1 Chiffrer ? Oui, mais comment ?

> *« [Prisme], il faut être certain que ça ne va pas s'arrêter, les Américains n'ont aucune raison de le démonter […]. Tout ce qu'on peut faire, au mieux, c'est de leur compliquer la vie, c'est-à-dire de chiffrer au max, de multiplier les points de stockage, de se méfier comme de la peste du Cloud. »*
>
> *Louis Pouzin – JRES 2013 [1]*

Comme le proclame Louis Pouzin[1], le chiffrement est indispensable. aujourd'hui, sa mise en œuvre est totalement indissociable des certificats X.509, dont le déploiement et la gestion présentent des écueils.

Cet article n'a aucunement la prétention de présenter des innovations, mais il rappelle les principes associés aux certificats dont les aspects techniques et les enjeux sont parfois mal maîtrisés ; il en découle alors des défaillances susceptibles de réduire à néant l'utilité des certificats.

2 Connaître les principes

2.1 Un peu de cryptographie

On traitera ici des processus fonctionnels dans lesquels entrent les algorithmes utilisés en cryptographie, pas de leur théorie, ce qui sortirait du cadre de cet article. Les trois principes qui suivent seront ensuite combinés (chiffrement hybride) pour construire des solutions pratiques, telles que les échanges via TLS, le mail confidentiel ou signé, ou la signature de code.

1. Inventeur du datagramme (projet Cyclades), élément fondamental d'IP, repris par Vinton Cerf lors de la conception de TCP/IP.

2.1.1 Le chiffrement symétrique

Le chiffrement symétrique permet de transformer un message en clair en sa version chiffrée, non compréhensible. Pour un même message d'entrée, le résultat, c'est-à-dire le message chiffré, varie en fonction d'une clé *k*, appelée clé de chiffrement.

Dans le cadre du chiffrement symétrique, c'est la même clé *k* qui sert au chiffrement et au déchiffrement.

Si f_k est la fonction de chiffrement et g_k celle de déchiffrement[2], on a (Figure 1) :

- au chiffrement : $X_k = f_k(x)$, avec x , message en clair, et X_k , message chiffré par la fonction f et la clé k ;
- au déchiffrement : $x = g_k(X_k)$
- avec $g_k(f_k) = f_k(g_k) = Id$

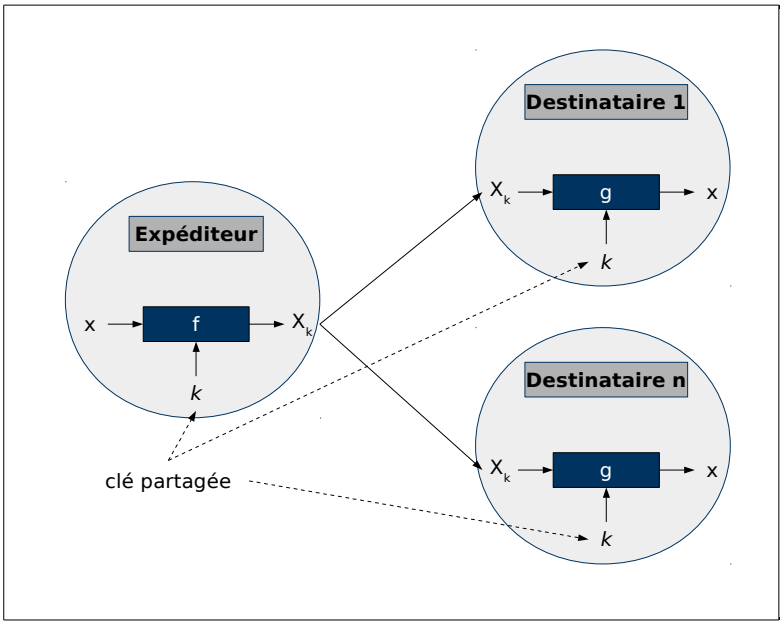

Figure 1 - Chiffrement symétrique d'un message x .

Si une telle fonction permet facilement à un utilisateur de dissimuler ses propres documents, on comprend rapidement que, pour un échange avec d'autres utilisateurs, il va falloir partager le secret de la clé de chiffrement avec l'ensemble des utilisateurs.

2.1.2 Le chiffrement asymétrique

Le chiffrement asymétrique permet de résoudre certains des problèmes non couverts par le chiffrement symétrique. Son concept a été défini en 1976 par les mathématiciens Diffie et Hellman puis mis en œuvre pour la première fois en 1978 par Rivest, Shamir et Adleman (RSA).

Le principe consiste à calculer un couple de clés de chiffrement, *p* et *s*, dont les propriétés sont :

- il est impossible de déduire l'une des clés même connaissant l'autre ;
- la propriété précédente implique que les clés doivent être choisies simultanément et que la perte de l'une d'elles rend le couple (*p*, *s*) totalement inutilisable ;
- la composition des fonctions de chiffrement correspond à la fonction identité : $f_p(f_s) = Id$;
- leur composition est commutative : $f_p(f_s) = f_s(f_p) = Id$.

Dans le cadre d'échanges, chaque utilisateur, personne physique ou morale, peut être pourvu d'un couple

2. Pour certains algorithmes, tels que DES, on peut avoir $f = g$.

de clés. L'une d'elle sera gardée confidentielle et représentera la clé privée (ou secrète *s*) et l'autre pourra être diffusée et représentera la clé publique (*p*) de l'utilisateur.

2.1.3 Scénarios types du chiffrement asymétrique

En appliquant les propriétés définies à la section précédente, on peut projeter trois scénarios types :

– authentification de A par B (Figure 2) : A utilise sa clé secrète S_A pour chiffrer et B utilise la clé publique P_A de A pour déchiffrer, car $f_{PA}(f_{SA})=Id$. Seul A était en mesure de chiffrer avec sa clé secrète, ce qui permet à B d'être certain de l'identité de A.

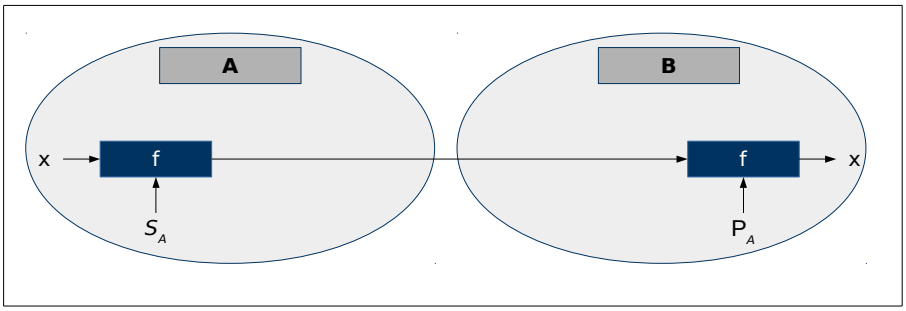

Figure 2 - Authentification de A par B.

– chiffrement par A pour B (Figure 3) : A utilise la clé publique P_B de B pour chiffrer et B utilise sa clé secrète S_B pouf déchiffrer, car $f_{SB}(f_{PB})=Id$. Seul B est en mesure de déchiffrer, ce qui permet à A d'être certain de la confidentialité.

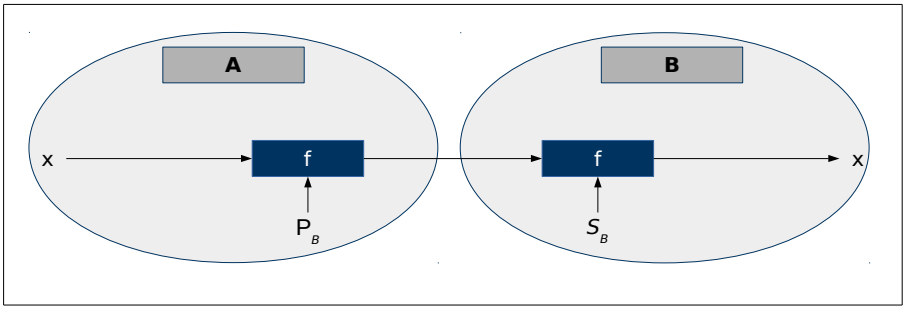

Figure 3 - Chiffrement par A pour B.

– authentification réciproque (Figure 4) : A utilise successivement sa clé secrète S_A puis la clé publique P_B de B pour chiffrer, et B utilise successivement sa clé secrète S_B puis la clé publique P_A de A pour déchiffrer, car $f_{PA}(f_{SB}(f_{PB}(f_{SA})))=f_{PA}(f_{SA})$, où $f_{SB}(f_{PB})=Id$ et $f_{PA}(f_{SA})=Id$, ce qui se résume au final à l'identité.

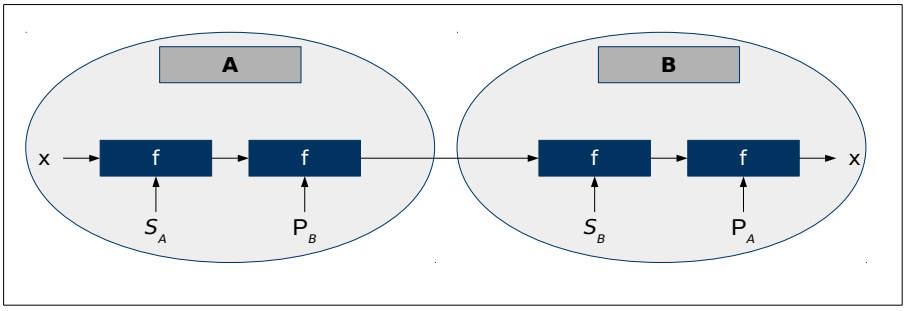

Figure 4 - Authentification réciproque de A et de B.

2.1.4 Empreinte

Une fonction de hachage permet d'obtenir une empreinte (ou condensat) à partir d'un document. Cette empreinte est de petite taille (quelques octets) quelle que soit la taille du document original, et généralement de longueur fixe. L'empreinte possède les propriétés suivantes :

– il est quasiment impossible de retrouver le document original à partir de l'empreinte ;

– connaissant le document original et l'empreinte, il est quasiment impossible de fabriquer un autre document ayant la même empreinte ;

– une modification légère du document original produit une modification massive de l'empreinte (effet d'avalanche).

En conclusion, on peut estimer que deux documents ayant la même empreinte sont en fait équivalents.

2.2 Les certificats

Le certificat X.509 est totalement orienté chiffrement asymétrique ; son rôle est d'associer l'identité d'une personne physique ou morale à sa clé publique, et de garantir cette association. Pour cela, le certificat est soumis à une autorité qui y appose sa signature. Par ailleurs, les utilisateurs des certificats devront faire confiance a priori à un ensemble d'autorités de certification.

Les certificats X.509 utilisés ici sont décrits dans le standard RFC 5280 de 2008 [2] par le groupe PKIX de l'IETF et sont une spécialisation de la norme X.509 de l'UIT-T.

2.2.1 Structure simplifiée

Un certificat est composé de champs et ne mesure que quelques kilo-octets (Figure 5).

Catégorie	Champ	Exemple / Commentaire
Propriétaire (sujet)	Nom	CN, OU, O, etc
	Clé publique	
	Algorithme des clés	Chiffrement PKCS #1 RSA
Autorité de Certification	Nom	CN, OU, O, etc
	Signature	
	Algorithme de signature	PKCS #1 SHA-256 avec chiffrement RSA
Certificat	Plage de validité	Dates de début et de fin
Extensions facultatives	Utilisation du certificat	Signature, chiffrement de la clé
	Noms alternatifs du sujet	
	Accès aux informations de l'AC	CRL, OCSP, annuaire

Figure 5 - Structure simplifiée d'un certificat.

Le champ « algorithme des clés » (*Subject Public Key Info*) indique ici que les clés publique et privée du propriétaire du certificat sont au format RSA.

Le champ « algorithme de signature » (*signatureAlgorithm*) de l'autorité de certification définit ici deux algorithmes de chiffrement, procédé appelé chiffrement hybride :

- SHA-256 est une fonction de hachage de la famille SHA-2, qui sert ici à prendre une empreinte de la totalité du certificat ;
- RSA est un algorithme de chiffrement asymétrique qui sert ici à signer l'empreinte calculée à l'étape précédente ;
- ce procédé permet de conserver de bonnes performances, ce qui ne serait pas le cas si les applications utilisatrices devaient signer l'ensemble du certificat par un algorithme asymétrique.

Enfin, remarquons que la clé privée de l'utilisateur ne fait pas partie du certificat. En effet, le certificat est une information publique et a vocation à être diffusé de manière large. Il existe des containers qui permettent d'intégrer le certificat et la clé privée associée, pour des besoins de transfert d'une application à une autre, tels que le format PKCS#12, ou pour être directement utilisé dans certaines applications, tels que le format JKS (*Java KeyStore*) des machines virtuelles Java.

2.2.2 Certification

Le certificat de l'utilisateur est signé par une autorité de certification (AC), elle-même détentrice d'un certificat pour attester de son identité et diffuser sa clé publique. Ce certificat, nommé certificat racine, est auto-signé.

Le chemin entre le certificat de l'utilisateur et le certificat racine n'est pas obligatoirement direct mais peut comporter plusieurs autorités de certification (AC) ou autorités d'enregistrement (AE) intermédiaires. Une autorité d'enregistrement n'est pas nécessairement en mesure d'émettre elle-même des certificats, mais elle reçoit par délégation un rôle de proximité avec l'utilisateur. Le certificat d'une AE intermédiaire se présente comme celui d'une AC, comme si l'AE avait elle-même signé le certificat du niveau inférieur.

La Figure 6 présente la hiérarchie, ou chemin de certification, dans le cadre du service de certificats TCS de Renater. Renater adhère au réseau européen Géant / Terena de la communauté de l'enseignement et de la recherche. Terena a conclut en 2015 un contrat avec l'AC DigiCert avec l'organisation suivante[3] :

- Terena sert d'autorité d'enregistrement (AE) pour ses membres ;
- le service d'AE est opéré directement par DigiCert.

Cette hiérarchie explique que lorsqu'un adhérent au service TCS demande un certificat, il s'adresse à l'AE Terena mais se connecte concrètement à l'AC DigiCert.

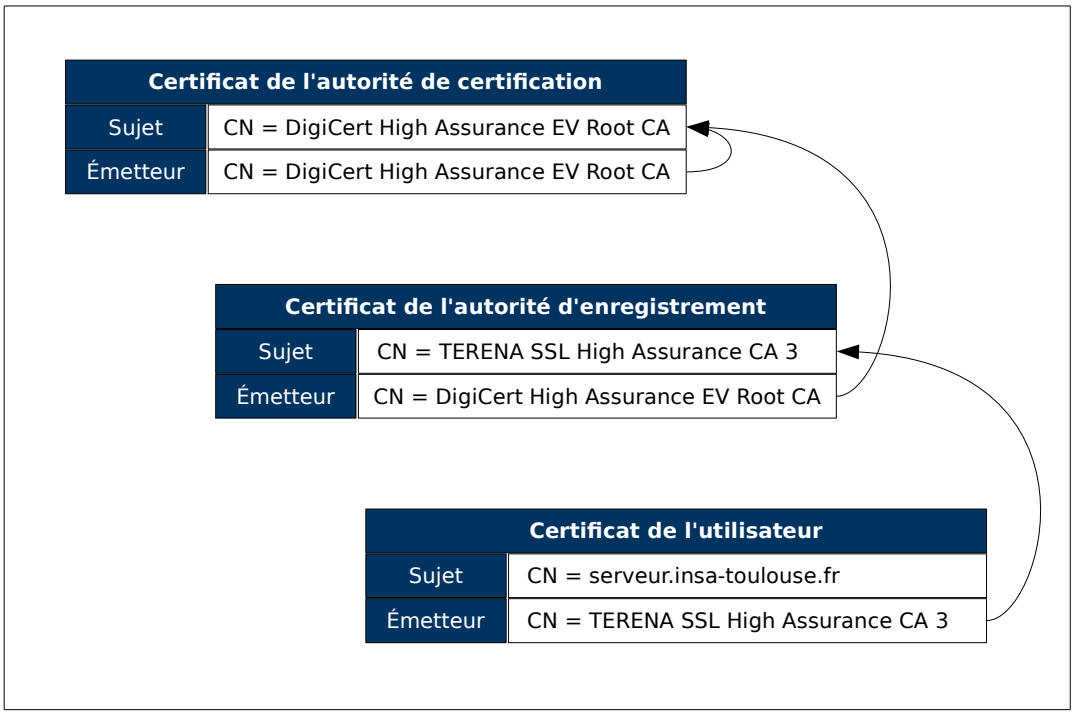

Figure 6 - Chemin de certification du service TCS de Renater / Terena pour un certificat EV.

Les rôles et procédures des autorités de certification et d'enregistrement sont décrites dans leur politique de certification. Ces documents constituent des engagements, auxquels l'utilisateur peut se référer pour effectuer ses choix après avoir déterminé ses besoins en sécurité.

3. De 2009 à 2015, Terena avait conclut un contrat avec l'AC Comodo.

À la différence du DNS, l'ensemble des autorités de certification ne forme pas une arborescence. L'utilisateur s'adresse à l'autorité de son choix pour demander la signature d'un certificat et pourrait tout à fait s'adresser à des autorités différentes d'un certificat à l'autre.

2.3 Application typique

Les certificats X.509 sont utilisés dans de nombreuses applications telles que les tunnels TLS (SSL), S/MIME, IPSec, SSH, signature de document ou de code. Les concepts étant globalement les mêmes (combinaison de chiffrement symétrique, asymétrique et de hachage), nous ne développerons que le cas de l'établissement d'un tunnel TLS entre un navigateur et un serveur HTTPS et en ne présentant que les étapes impliquant le certificat lui-même.

Comme nous l'avons vu dans la section précédente, il n'existe pas de système global reliant les autorités de certification entre elles. Ainsi, un utilisateur de certificat ne peut a priori pas trouver le certificat d'un tiers à partir d'une identité. Il se pose donc le problème de la propagation du certificat et, implicitement, de la clé publique. Ce rôle est en fait dévolu aux applications utilisatrices elle-mêmes car, dans la grande majorité des cas, la propagation d'un certificat est intégrée au processus métier de l'application comme, par exemple, pour :

- connexion TLS : le certificat est échangé lors de l'étape de négociation, y compris dans le cas d'une authentification réciproque service ↔ client ;
- S/MIME : le certificat est transmis en pièce jointe avec le message ;
- signature de code : dans le cas d'une applet Java, le certificat est inclus dans le fichier JAR ;
- enfin, il serait aussi possible de s'échanger préalablement les certificats « de la main à la main ».

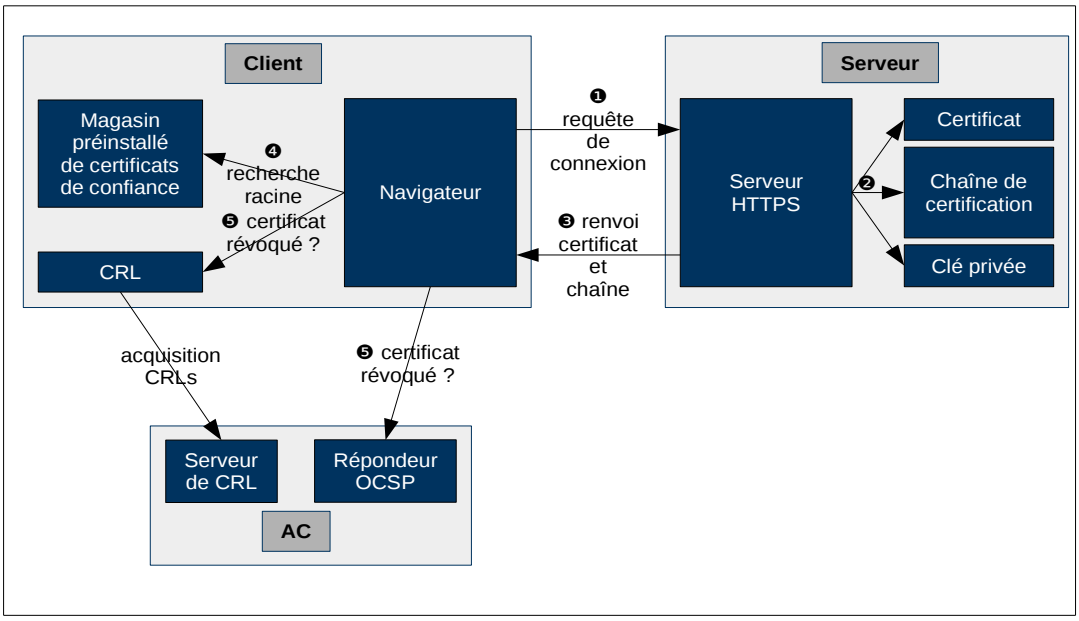

Figure 7 - Implication d'un certificat dans l'établissement d'un tunnel TLS entre un serveur HTTPS et un navigateur.

Le client initie la connexion. Le serveur lui renvoie, entre autres, son certificat et sa chaîne de certification.

Le client doit ensuite s'assurer de l'identité du serveur, via deux opérations de validation du certificat reçu :

- sur le certificat en lui-même :
  - chaîne de certification : chaînage complet, habilitation de chaque autorité à émettre un certificat pour l'usage d'authentification, présence du certificat racine de la chaîne dans le

magasin local de certificats a priori reconnus et pré-installés sur le client ;
    - acceptabilité des algorithmes de signature et des tailles de clé ;
    - validation de la signature de chaque certificat ;
    - vérification des dates de validité de chaque certificat ;
- sur sa validité à l'instant présent, c'est-à-dire l'absence de révocation du certificat à l'initiative de son propriétaire ou de l'AC, avant sa date de fin de validité :
    - absence de l'ensemble des certificats des listes de révocation (ou CRL, *Certificate Revocation List*) dont la méthode d'accès est décrite dans les certificats eux-mêmes ;
    - réponse positive à une interrogation directe de l'autorité émettrice concernée, dont l'adresse est définie dans le certificat, à l'aide du protocole OCSP (*Online Certificate Status Protocol*).

Si l'un des contrôles échoue, l'application cliente fait en principe appel à l'utilisateur pour une prise de décision.

3 Quelques bonnes pratiques

3.1 Cryptographie : algorithmes et longueur de clés

Le problème qui nous intéresse ici porte aussi bien sur les algorithmes et la longueur des clés véhiculés par le certificat pour être utilisés dans les processus métier, que ceux utilisés pour la signature du certificat.

Leur choix dépendra à la fois de la capacité de prise en charge des applications métier et de l'état de l'art en cryptographie. Les besoins ne sont pas non plus les mêmes d'un processus métier à l'autre : authentification de connexion, signature d'un document à pérennité courte ou longue, etc. Pour l'ensemble de ces raisons, il est recommandé de répartir les usages sur des certificats différents. Ainsi, par exemple, une personne physique pourrait disposer d'un certificat pour l'authentification, d'un autre pour la signature et le chiffrement de mail, d'un autre encore pour la signature de document.

Recommandation : utiliser des certificats différents pour la signature, l'authentification…

Actuellement, en 2015, l'état de l'art nous alerte sur l'algorithme de hachage SHA-1 dont la fin déjà annoncée s'est rapprochée plus rapidement que prévu. SHA-1 a longtemps été utilisé pour prendre l'empreinte du certificat et la chiffrer ensuite avec la clé privée de l'autorité de certification, le résultat constituant la signature du certificat. Si on observe le service TCS (voir section 2.2.2), on constate que les certificats SSL/TLS générés par Comodo jusqu'à la mi-2014 sont signés en utilisant SHA-1. Après mi-2014 pour Comodo, et avec le nouveau prestataire DigiCert, l'empreinte des certificats est prise avec la famille SHA-2 (SHA-256). Mais certains des certificats TCS générés jusqu'à la mi-2014 seront encore valides jusqu'à la mi-2017, si on tient compte de la durée de validité maximale de 3 ans qui pouvait être demandée. Il est conseillé de révoquer et remplacer ces certificats sans attendre la fin de leur validité. Un outil en ligne tel que https://shaaaaaaaaaaaa.com/ permet de tester la présence de SHA-1 sur un serveur exposé sur Internet.

Figure 8 - Exemple d'échec, puis de réussite, au test par https://shaaaaaaaaaaaa.com/.

Recommandation : remplacer les certificats serveur signés par une empreinte SHA-1.

De manière générale, il faut se baser sur l'état de l'art pour effectuer les choix de configuration. Une bonne piste est de suivre les recommandations du RGS, même pour les services ne relevant pas de l'homologation RGS, en se référant à l'annexe B-1 « Mécanismes cryptographiques » du RGS 2.0 [3] dans laquelle des abaques donnent des indications sur la longueur des clés selon la durée de validité nécessaire par rapport aux processus métier à mettre en œuvre.

3.2 Protection des clés privées

La protection des clés privées doit être conforme aux conclusions des analyses de risque.

Dans le cas de service, la clé privée ne devrait être accessible qu'à un nombre très limité de personnes privilégiées, ce qui peut être parfois difficile à gérer lorsque le serveur est multi-fonction. Par ailleurs, il faut protéger la clé privée d'un éventuel vol. Les accès au service doivent donc être correctement gérés, et les procédures doivent être respectées en permanence, sous peine de risquer une fuite de clé privée, comme cela est arrivé en février 2015 chez D-Link, qui a publié une clé privée dans des paquetages applicatifs durant plusieurs mois avant de révoquer le certificat en septembre 2015.

> Recommandation : limiter les accès aux serveurs comportant des clés privées ; centraliser les services sur des équipements maîtrisés par le service informatique.

Dès que l'on ne peut pas maîtriser suffisamment l'accès privilégié (délégation partielle d'administration), il est nécessaire d'utiliser une enceinte cryptographique. Dans un tel équipement matériel, la clé privée n'est jamais exposée.

> Recommandation : utiliser autant que possible une enceinte de cryptographique[4].

Dans le cas d'un certificat de personne, la protection de la clé privée est beaucoup plus délicate. En effet, le couple certificat / clé privée doit pouvoir être manipulé simultanément (par exemple dans un container PKCS#12) pour être intégré dans différentes applications. L'idéal pour protéger la clé privée est là aussi de disposer d'un composant matériel (*token* USB, carte à puce, etc.), mais cela déplace alors la difficulté sur la mise en œuvre de ce composant.

3.3 Chaîne de certification et certificats auto-signés

Une chaîne de certification mal construite et un certificat auto-signé mènent au même résultat : l'impossibilité pour l'utilisateur de s'assurer de la validité du certificat. Malgré l'alerte de sécurité, la plupart des utilisateurs, souvent par manque de sensibilisation, acceptent la poursuite de l'utilisation de l'application (lorsque celle-ci le permet), le pire étant que cette acceptation de certificat non vérifié peut devenir tacite. Même en prévenant un utilisateur qu'il est normal qu'il doive accepter le certificat d'un service connu à certificat auto-signé, ce serait un très mauvais signal qu'on lui donne, car l'utilisateur aura tendance à reproduire cette acceptation de manière systématique. Un utilisateur ne devrait jamais avoir à accepter un certificat, et le message qu'on lui donne doit être simple et constant.

Heureusement, les éditeurs d'application commencent à prendre en compte ce problème et empêchent d'outrepasser l'alerte de sécurité. C'est le cas, par exemple, avec Java depuis la version 7 parue à l'automne 2013 : tout applet Java chargé par *Web Start* doit être signé par un certificat reconnu. A contrario, d'autres fournisseurs de service proposent encore de pré-charger le certificat auto-signé de leur service dans le magasin de certificats de l'utilisateur, ce qui revient, en quelques sortes, à faire de ce certificat un certificat racine.

La chaîne de certification doit souvent répondre à des critères précis tels que l'ordre des certificats intermédiaires. La JVM de Oracle/Sun est particulièrement sensible à cet ordre : ainsi, si le service de communication sécurisé est rendu par une application Java (serveur CAS de Jasig par exemple), le certificat serveur, la clé privée et la chaîne de certification doivent être introduits en une unique opération et dans le bon ordre dans le magasin de la JVM (JKS ou *JavaKeystore*) du serveur, ce qui n'est pas une opération banale lorsque la demande de certificat n'a pas été générée par la JVM elle-même.

4. Enceinte cryptographique : équipement matériel offrant des services de chiffrement à un ensemble de serveurs et isolant les clés privées dans des zones mémoire inaccessibles de l'extérieur.

Recommandation : s'assurer de fournir des certificats signés par une autorité reconnue et des chaînes de certification correctes, afin d'éviter que les utilisateurs ne soient amenés à accepter une exécution malgré des messages d'alerte.

3.4 Certificats racine

Lorsqu'une application client doit valider le certificat reçu, elle remonte la chaîne de certification et aboutit finalement à un certificat racine auto-signé. Pour achever la validation, l'application cliente recherche la présence de ce certificat auto-signé dans son magasin de certificats de confiance.

Ces certificats de confiance sont généralement livrés avec le système d'exploitation ou l'application cliente : répertoire */etc/ssl/certs* de OpenSSL, JavaKeystore *lib/security/cacerts* d'une JVM. Ils sont même parfois intégrés au code de l'application (Mozilla Firefox et Thunderbird).

Or ces certificats ont, comme tout certificat, une durée de validité limitée :

- leur fin de validité normale est bien sûr indiquée dans le certificat lui-même ;
- ils peuvent être révoqués prématurément à l'initiative de l'autorité de certification elle-même, soit suite à une compromission, soit pour des raisons d'évolution de l'état de l'art en cryptographie ;

Enfin, une autorité de certification peut ne plus être digne de confiance. Cette dernière situation s'est produite à plusieurs reprises ces dernières années :

- en 2011, les autorités de certification Comodo et Diginotar ont été compromises et des certificats ont été émis par des pirates (ce qui a conduit à la faillite de Diginotar) ;
- par manque de respect des procédures, des faux certificats ont été émis par les employés de certaines AC et ont circulé sur Internet (certificats Google émis par l'IGC/A en 2013 ou par une AC chinoise en 2015) ;
- dans tous ces cas, les certificats racine de ces autorités ont été retirés, au moins momentanément, des magasins de certificats de confiance par les éditeurs d'application.

Bien sûr, le flux des autorités de certification de confiance et de l'ensemble des certificats de leurs différentes branches ne peut être répercuté que si les applications intégrant des magasins et déjà déployées sont mises à jour régulièrement. Dans certains cas, il peut s'avérer difficile de tenir à jour l'ensemble des magasins, par exemple si l'on doit faire cohabiter plusieurs versions différentes d'une même application, ce qui est parfois le cas avec les machines virtuelles Java (nécessité fonctionnelle, par exemple, de garder en service une JVM 1.6 ou 7 alors qu'elles ne sont plus maintenues par l'éditeur).

Recommandation : tenir à jour les applications intégrant un magasin de certificats racine.

3.5 Certificats *wildcard*

Le certificat omni-domaine ou *wildcard* est défini dans la RFC 2818. C'est un certificat pour serveur TLS dont le nom du sujet a la forme **.insa-toulouse.fr* et qui peut donc fonctionner pour tout service dont le nom du sujet a la même forme. Un service ayant un tel certificat peut donc se substituer à tout service ayant un certificat spécifique, et si la clé privée associée à un certificat *wildcard* est compromise, l'ensemble de la sécurité de l'organisme l'est également. Il est donc évident que la clé privée associée doit être particulièrement protégée, idéalement par l'utilisation d'une enceinte cryptographique dans laquelle la clé secrète est générée en interne.

La simple existence d'un tel certificat est un risque et il est nécessaire d'assurer fortement la protection de la clé secrète durant toute la durée de validité du certificat. Le mieux est même d'éviter de générer un certificat *wildcard* et de le réserver à des situations où il présente un intérêt fort, tel qu'un service de création dynamique de sites web.

Recommandation : éviter la génération de certificat *wildcard* si ce n'est pas indispensable.

3.6 Vérification de l'état d'un certificat

Dans la section 2.3, nous avons vu qu'une application peut vérifier la non-révocation d'un certificat avant la fin de sa validité en chargeant une liste de révocation de certificat à partir d'un serveur de CRL, ou en interrogeant un serveur OCSP pour un certificat précis. Aucun standard ne précise comment une application doit enchaîner les méthodes de vérification, en particulier si elle doit, ou non, utiliser toutes les méthodes (CRL ou OCSP) à sa disposition, et dans quel ordre. Dans tous les cas, l'application trouvera les points de contact sous la forme d'URI dans les champs d'extension du certificat à vérifier. Ces points de contacts sont accessibles en HTTP ou HTTPS, généralement sur les ports standard. Pour que les vérifications puissent aboutir, le client doit être en mesure de contacter les serveurs de CRL ou les serveurs OCSP. En général, l'absence de réponse n'empêche pas l'application de poursuivre son travail.

Recommandation : ne pas bloquer le trafic vers les serveurs de CRL ou les serveurs OCSP.

3.7 Convergence autorité de certification / application

Au final, le certificat n'aura de la valeur que s'il est bien géré par l'application. Rien ne servirait à une autorité de certification de mettre seule en place des pratiques innovantes, si celles-ci n'étaient pas connues des applications.

Pour cette raison, des autorités de certification et des éditeurs de navigateur (Apple, Microsoft, Mozilla, Opera) se sont regroupés pour s'accorder sur des bonnes pratiques, au sein du groupe CAB, ou CA/Browser Forum (www.cabforum.org). L'activité principale de ce groupe est actuellement de promouvoir les certificats EV (*Extended Validation*) dans les navigateurs. Le groupe travaille aussi sur les certificats EV de signature de code.

En ce qui concerne les autorités de certification, le CA/Browser Forum définit les procédures à mettre en œuvre avant de générer un certificat de l'une des catégories suivantes :

- DV (*Domain Validation*) : l'AC s'assure que la demande de signature provient bien du propriétaire du domaine utilisé dans le DN (*Distinguished Name*) du certificat, en s'appuyant sur un échange de mail d'après les informations contenues dans le WHOIS du domaine ;
- OV (*Organizational Validation*) : l'AC valide également le nom de l'organisme (champ O du certificat) en consultant des bases de données publiques ;
- EV (*Extended Validation*) : le contrôle par l'AC est encore plus strict, généralement par un appel téléphonique au numéro public de l'organisme, interrogation du service RH pour vérifier que la personne ayant demandé le certificat est bien membre de l'organisme et qu'elle occupe bien la fonction indiquée, enfin contact téléphonique avec le demandeur.

En ce qui concerne les navigateurs, l'accès à un serveur en HTTPS dont le certificat est de type EV, se traduit par un affichage en vert au début de la barre d'adresse.

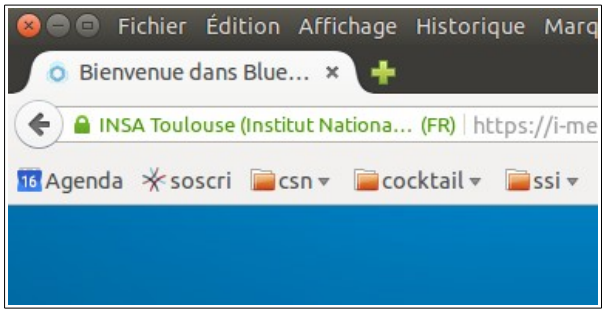

Figure 9 - Affichage du nom officiel de l'organisme avec un certificat EV.

	Autorité de certification			Navigateur	
Type de certificat	Validation du domaine	Validation du nom du sujet	Validation de l'adresse postale	Affichage d'un cadenas	Affichage en vert dans la barre d'adresse
DV	X			X	
OV	X	X	X	X	
EV	X	X	X	X	X

Figure 10 - Prise en compte des certificats de type DV, OV et EV par les AC et les navigateurs.

L'affichage propre aux certificats EV dans le navigateur garantit à l'utilisateur :

- non seulement que le nom de domaine visité appartient bien au propriétaire du domaine (comme avec tout certificat DV) ;
- mais aussi que le site visité vient bien de l'entreprise dont le nom est indiqué en vert (EV). En obligeant l'ensemble des services web de l'organisme à fonctionner en HTTPS et en utilisant exclusivement des certificats EV, il est possible de communiquer avec des messages simples auprès de son propre personnel : « si vous pensez vous connecter à un service de l'organisme et que le début de la barre d'adresse n'affiche pas notre nom officiel en vert, alors vous pouvez être certain que le site n'est pas de chez nous ». Le message est simple et il évite de devoir conseiller des comportements différents selon la situation. En outre, il peut probablement contribuer à lutter contre le hameçonnage au « service informatique ».

  Recommandation : utiliser systématiquement des certificats EV pour les services web.

Notons enfin que les éditeurs de navigateur envisagent de bannir le trafic HTTP au profit de HTTPS (voir « HTTPS as a ranking signal » par Google ou « Deprecating Non-Secure HTTP » par Richard Barnes, responsable de la sécurité de Firefox). Il est donc temps d'anticiper la migration intégrale vers HTTPS.

Recommandation : migrer la totalité des services web en HTTPS.

Conclusion

Voici donc remémorés les concepts de base. Les exploitants que nous sommes y trouveront des directions pour de bonnes pratiques.

Malheureusement, la liste des problèmes exposés est loin d'être exhaustive. Certaines de ces difficultés ont déjà défrayé nombre de colloques JRES et, malgré leur ancienneté, elles restent parfois une préoccupation d'actualité. Face à cela, il est indispensable de continuer à déployer les certificats X.509 en accomplissant du mieux possible la partie qui nous incombe.

L'un de ces problèmes, et peut-être le plus fondamental, est qu'il n'y a pas moyen de savoir si le sujet réel défini dans le certificat est bien client de l'AC qui a signé ce certificat (voir [15]), et c'est le dossier sur lequel il va falloir se pencher maintenant.

Bibliographie

[1] Pouzin L., CYCLADES : les premières bases d'un réseau de réseaux, JRES 2013, Montpellier, http://video.renater.fr/jres/2013/index.php?aid=184

[2] Cooper S., Santesson S., Farell S., Boeyen S., Housley R. et Polk W., RFC 5280, Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, 2008, https://tools.ietf.org/html/rfc5280

[3] ANSSI, Mécanismes cryptographiques, Règles et recommandations concernant le choix et le dimensionnement des mécanismes cryptographiques, RGS 2.0, Annexe B1, 2014, http://www.ssi.gouv.fr/uploads/2015/01/RGS_v-2-0_B1.pdf

[4] Aumont S., Dirlewanger R. et Porte O., L'accès sécurisé aux données, JRES 1999, Montpellier, http://1999.jres.org/tutoriaux/tutorial4-chiffrement.pdf

[5] Aumont S., Gross C. et Leca P., Certificats X509 et Infrastructure de Gestion de Clés, JRES 2001, Lyon, https://www.cru.fr/igc/JRES01.tutoriel.IGC.pdf

[6] Archimbaud J.L., Mise en place progressive d'une IGC au CNRS, JRES 2001, Lyon, http://2001.jres.org/actes/igcaucnrs.pdf

[7] Aumont S., S/MIME et le gestionnaire de listes de diffusion Sympa, JRES 2001, Lyon, http://2001.jres.org/actes/smime.pdf

[8] Frausto Bernal P.A., Antoine C. et Serhouchni A., Utilisations des certificats d'attribut pour accélérer l'usage de la signature électronique, JRES 2003, Lille, , http://2003.jres.org/actes/paper.45.pdf

[9] Autret T., Bellefin L. et Oble-Laffaire M,L., Sécuriser ses échanges électroniques avec une PKI, Eyrolles, 2002

[10] Cachat C. et Carella D., PKI Open Source, O'Reilly, 2003

[11] Aumont S., Faut-il brûler vos certificats ?, JRES 2003, Lille, http://2003.jres.org/actes/paper.21.pdf

[12] Parouty J.L., Dirlewanger R. et Vaufreydaz D., La signature électronique, contexte, applications et mise en œuvre, 2003, http://2003.jres.org/actes/paper.79.pdf

[13] Herrb M., Sécurisation de la messagerie électronique, JRES 2003, Lille, http://2003.jres.org/TUTORIELS/paper.B.pdf

[14] Guezou J.F., Launay D. et Aumont S., SCS est mort, vive TCS !, JRES 2009, Nantes, https://2009.jres.org/planning_files/article/pdf/76.pdf

[15] Bortzmeyer S., Des clés dans le DNS, un successeur à X.509 ?, JRES 2011, Toulouse, https://2011.jres.org/archives/167/paper167_article.pdf

[16] Prévost E., Plate-forme mutualisée de signature électronique, JRES 2013, Montpellier, https://conf-ng.jres.org/2013/planning.html#article_142

[17] Ristic I., OpenSSL Cookbook, Feisty Duck, 2015, https://www.feistyduck.com/books/openssl-cookbook/