LES GUIDES DE
MISC
Multi-System & Internet Security Cookbook
HORS-SÉRIE
N°13

France MÉTRO. : 12,90 € — CH : 18,00 CHF — BEL/PORT.CONT : 13,90 € — DOM TOM : 13,90 € — CAN : 18,00 $ CAD

VIE PRIVÉE, E-COMMERCE, SÉCURITÉ

CRYPTOGRAPHIE

LES ÉLÉMENTS CLÉS POUR MIEUX COMPRENDRE SES PRATIQUES & SES ENJEUX !

LE CHIFFREMENT POUR TOUS
Découvrez les protocoles et les projets qui veulent vous protéger

LES TIERS DE CONFIANCE EN QUESTION
Les dernières évolutions qui bouleversent le monde des autorités de certification

L'ENJEU DE LA NORMALISATION
Découvrez les derniers standards qui occuperont une place importante dans l'écosystème de demain

LES AVANCÉES D'AUJOURD'HUI ET DE DEMAIN
Initiez-vous aux courbes elliptiques et à l'informatique quantique

Édité par Les Éditions Diamond
L 16844 - 13 H - F: 12,90 € - RD
www.ed-diamond.com

Retrouvez toutes nos publications
LES ÉDITIONS DIAMOND
sur **www.ed-diamond.com**

MISC Hors-Série
est édité par **Les Éditions Diamond**

10, Place de la Cathédrale – 68000 Colmar – France

Tél. : 03 67 10 00 20 / **Fax :** 03 67 10 00 21

E-mail : cial@ed-diamond.com

Service commercial : abo@ed-diamond.com

Sites : www.miscmag.com
www.ed-diamond.com

Directeur de publication : Arnaud Metzler

Chef des rédactions : Denis Bodor

Rédacteur en chef : Cédric Foll

Secrétaire de rédaction : Aline Hof

Conception graphique : Kathrin Scali

Remerciements à gapz

Responsable publicité : Tél. : 03 67 10 00 27

Service abonnement : Tél. : 03 67 10 00 20

Impression : pva, Druck und Medien-Dienstleistungen GmbH, Landau, Allemagne

Distribution France :
(uniquement pour les dépositaires de presse)

MLP Réassort :
Plate-forme de Saint-Barthélemy-d'Anjou.
Tél. : 02 41 27 53 12
Plate-forme de Saint-Quentin-Fallavier.
Tél. : 04 74 82 63 04

Service des ventes :
Abomarque : 09 53 15 21 77

IMPRIMÉ en Allemagne - PRINTED in Germany

Dépôt légal : A parution

N° ISSN : 1631-9036

Commission Paritaire : K 81190

Périodicité : Bimestrielle

Prix de vente : 12,90 Euros

Membre April
www.april.org

Les articles non signés contenus dans ce numéro ont été rédigés par les membres de l'équipe rédactionnelle des Éditions Diamond.

CHARTE DE MISC :

MISC est un magazine consacré à la sécurité informatique sous tous ses aspects (comme le système, le réseau ou encore la programmation) et où les perspectives techniques et scientifiques occupent une place prépondérante. Toutefois, les questions connexes (modalités juridiques, menaces informationnelles) sont également considérées, ce qui fait de MISC une revue capable d'appréhender la complexité croissante des systèmes d'information, et les problèmes de sécurité qui l'accompagnent. MISC vise un large public de personnes souhaitant élargir ses connaissances en se tenant informées des dernières techniques et des outils utilisés afin de mettre en place une défense adéquate. MISC propose des articles complets et pédagogiques afin d'anticiper au mieux les risques liés au piratage et les solutions pour y remédier, présentant pour cela des techniques offensives autant que défensives, leurs avantages et leurs limites, des facettes indissociables pour considérer tous les enjeux de la sécurité informatique.

PRÉFACE

« *It is poor civic hygiene to install technologies that could someday facilitate a police state.* » - Bruce Schneier

Pour ce hors-série dédié à la cryptographie, nous avons décidé de traiter la thématique en prenant comme angle d'attaque les progrès qui vont effectivement toucher le grand public. De nos jours, il ne fait plus de doute que la sécurité se compose d'un ensemble d'éléments et qu'une attention particulière doit être donnée à chacun d'eux.

Cependant, la situation politique actuelle, outre-Atlantique comme en Europe, voit se durcir le débat et la législation qui entoure les moyens cryptographiques (en témoigne la symbolique opposition d'Apple à la requête faite par le FBI d'accéder au contenu d'un téléphone de leur fabrication). De la même manière, la surveillance des moyens de communication électronique prend une ampleur jamais connue par le passé. Utiliser correctement la cryptographie a désormais un impact grandissant sur la sécurité, d'autant plus si celle-ci est déployée massivement.

Quand bien même le lecteur passionné aura son mot à dire sur les attaques par canaux auxiliaires, les erreurs d'implémentation et bien d'autres détails d'importance, nous avons pris le choix délibéré de faire un numéro accessible. L'administrateur système ou l'initié trouveront donc des pistes sur la manière d'utiliser la cryptographie, mais également de quoi en comprendre les enjeux actuels.

Ainsi, ce numéro est organisé de la manière suivante : une première partie traite d'OTR (chiffrement de bout en bout et confidentialité persistante) ainsi que de projets visant à démocratiser de bons usages de la cryptographie. Une seconde partie aborde les dernières avancées notables du monde des autorités de certification. Seront ensuite présentées les dernières normes qui visent à être massivement déployées (les nouvelles courbes elliptiques dans TLS, et une API crypto pour les navigateurs web) pour enfin, dans la dernière partie, introduire les courbes elliptiques et les enjeux actuels de la cryptographie post-quantique.

Bonne lecture !

gapz

Sommaire

MISC Hors-Série

N°13

CRYPTOGRAPHIE

3 L'ENJEU DE LA NORMALISATION

4 LES AVANCÉES D'AUJOURD'HUI ET DE DEMAIN

INTRODUCTION

gapz

Ces dernières années auront vu naître un intérêt croissant de la cryptographie à destination du grand public. Du développement de protocoles destinés à la protection de la vie privée en passant par la normalisation de courbes elliptiques réputées « de confiance », il apparaît aujourd'hui intéressant d'étudier quelques-unes de ces initiatives qui ont vocation à répondre à des problèmes concrets posés par l'utilisation de plus en plus généralisée de la cryptographie.

Le hors-série que vous tenez entre vos mains traitera ainsi de sujets comme le développement de nouveaux protocoles et l'élaboration de nouveaux standards, mais également de projets à la frontière entre l'administration système et la cryptographie.

Vous retrouverez ainsi les articles organisés autour de quatre thèmes principaux, dans l'ordre : des projets visant à défendre la vie privée, les autorités de certification, les normes, les primitives actuelles et à venir.

La première partie, bien qu'étant hétérogène dans les sujets traités, a un dénominateur commun : chacun des projets présentés (OTR, bettercrypto, et un article non technique sur les cafés vie privée) se soucie des problématiques concrètes posées par la surveillance généralisée. Ainsi, OTR cherche par exemple à améliorer la confidentialité des communications instantanées en prenant également en compte l'utilisabilité (en permettant une authentification mutuelle plus simple avec le protocole SMP), bettercrypto s'attache à faciliter le travail des administrateurs système quant au choix des primitives à utiliser (pour avoir de la confidentialité persistante et supprimer les primitives obsolètes par exemple) et l'initiative café vie privée vise tout simplement à populariser certains usages de la cryptographie auprès du grand public.

De manière beaucoup plus classique, la suite du numéro traite de quelques projets mettant plus ou moins en mouvement le monde des autorités de certification : d'abord, en permettant à quiconque d'obtenir un certificat gratuitement avec le projet let's encrypt, en essayant ensuite d'améliorer le système de réputation des autorités de certification avec « certificate transparency » et, enfin, en permettant de détecter simplement une interception de trafic TLS.

Dans la troisième partie, il sera traité d'un sujet aux enjeux essentiels pour la cryptographie : la normalisation. Tout d'abord, le développement par le W3C d'une norme pour une interface de programmation pour la cryptographie dédiée aux navigateurs sera étudié. Notons que faire tourner du JavaScript implémentant de la cryptographie directement dans un navigateur pourrait grandement faciliter son utilisation, mais va également introduire de nouveaux risques. Ensuite, seront abordés et discutés les différents choix récents de courbes elliptiques qui ont été normalisées par l'IETF au regard des problèmes de confiance qu'a pu faire éclater l'histoire de la porte dérobée du standard Dual_EC_DRBG.

Enfin, quelques primitives actuelles et celles à venir seront introduites pour ceux qui s'intéressent de plus près au fonctionnement de la cryptographie. Premièrement, les courbes elliptiques, avec une approche orientée sur les applications pratiques. Leur intérêt ne fait aucun doute : pour des tailles de clef plus petites et des calculs plus rapides, nous pouvons obtenir un niveau de sécurité identique à celui de RSA (pour des tailles de clef bien plus grandes avec ce dernier bien sûr). Aussi, même s'il y a eu des avancées intéressantes au niveau mathématique, les attaques contre ces dernières n'ont pas connu d'amélioration significative ces dernières années. Enfin, pour clôturer ce numéro, le sujet passionnant de la cryptographie post-quantique sera abordé. Depuis l'annonce par la NSA de l'abandon de ses recommandations (la suite B, essentiellement basée sur les courbes elliptiques), l'intérêt du développement de primitives résistantes à de futures attaques opérées par des ordinateurs quantiques a refait surface. Le NIST vient donc, en ce début d'année 2016, de lancer la course au standard de cryptographie post-quantique.

Bonne lecture ! ▮

APPS CLOUD READY!*

CLOUD SPECTATOR
NUMÉRO 1
Meilleur rapport qualité/prix
Test comparatif 01/2016

intel Powered by Cloud Technology

1&1 SERVEUR CLOUD
1 MOIS GRATUIT !*

1&1 Cloud App Center est le meilleur moyen de lancer vos applications ! Choisissez parmi plus de **100 applications ultra-modernes** et associez-les à la vitesse et aux performances du serveur Cloud 1&1, **numéro 1 du test comparatif Cloud Spectator !**

- ✓ **Plateforme puissante et sécurisée**
- ✓ **Pas besoin de connaissance en serveurs**
- ✓ **Facturation à la minute**

1 MOIS POUR ESSAYER
1 CLIC POUR CHANGER DE PACK
1 APPEL UN EXPERT VOUS RÉPOND

0970 808 911
(appel non surtaxé)

1&1
1and1.fr

*Prêt pour le Cloud. 1&1 Serveur Cloud : 1 mois d'essai gratuit, puis à partir de 4,99 € HT/mois (5,99 € TTC) (pour la configuration du serveur Cloud S). Facturation mensuelle en fonction de la configuration choisie. Pas de durée minimum d'engagement. Des frais de mise en service de 9,99 € HT (11,99 € TTC) s'appliquent. Conditions détaillées sur 1and1.fr. Intel® et le logo Intel® sont des marques commerciales d'Intel Corporation aux États-Unis et/ou dans d'autres pays. 1&1 Internet SARL, RCS Sarreguemines B 431 303 775.

1

1 LE CHIFFREMENT POUR TOUS

À découvrir dans cette partie...

1.1 OTR, le chat chiffré garanti sans traîtrise

OTR est un protocole destiné à offrir les bonnes propriétés cryptographiques lorsque l'on souhaite communiquer de manière privée. Découvrez sur quelles bases il repose ainsi que ses possibles évolutions futures. p. 12

1.2 Des cryptoparties aux salons vie privée

Après les célèbres « crypto-party », où l'on échangeait des fingerprints PGP, découvrez comment celles-ci ont évolué au fil des années pour s'ouvrir au grand public et tenter de répondre aux problématiques de la surveillance de masse. p. 24

1.3 BetterCrypto.org : guide de cryptographie appliquée

Il est parfois compliqué de s'orienter dans la grande variété de primitives cryptographiques disponibles pour configurer TLS pour son serveur HTTPS, son VPN, ou n'importe quel autre service utilisant la cryptographie. Voici un projet qui tente de résoudre ce problème. p. 34

OTR, LE CHAT CHIFFRÉ GARANTI SANS TRAÎTRISE

Julien VOISIN

OTR existe depuis plus de dix ans, et permet à tout le monde de protéger ses échanges instantanés, non seulement d'attaquants externes, mais aussi d'un interlocuteur mal intentionné.

À la fin de son documentaire *Citizenfour* [1], Laura Poitras remercie plusieurs personnes, mais également plusieurs logiciels : TrueCrypt, Tor, GnuPG, OTR... En effet, ce sont eux qui ont été utilisés pour établir et maintenir une communication entre elle et le lanceur d'alerte Edward Snowden, pendant la transmission de documents top secret, à l'insu de la NSA. Ce n'est pas un hasard, car justement, un de ces documents mentionne le fait que ces outils causent des « problèmes majeurs » aux agences de surveillance.

Pourtant, si aujourd'hui tout le monde connaît les trois premiers (Truecrypt pour le chiffrement de disques, Tor pour l'anonymat, GnuPG pour chiffrer les courriels), OTR reste relativement méconnu, ce qui est bien dommage, car il offre des possibilités uniques et très intéressantes pour s'assurer que des conversations secrètes le soient, et le restent, y compris dans des situations ubuesques.

1. OTR, MAIS ENCORE ?

OTR signifie *Off the record*, un terme journalistique d'origine anglo-saxonne se référant à la règle dite de *Chatham House*, énonçant que les participants sont libres d'utiliser les informations, mais qu'ils ne doivent révéler ni l'identité, ni l'affiliation des personnes à l'origine de ces informations, de même qu'ils ne doivent pas révéler l'identité des autres participants. C'est précisément l'une des fonctionnalités d'OTR : garantir que dans un échange, il ne sera pas possible de lier la conversation à son interlocuteur, tout en étant certain que c'est bien à lui que l'on s'adresse durant celle-ci.

OTR a été présenté en 2004 par Nikita Boris, Ian Goldberg et Éric Brewer dans leur papier *Off-the-Record Communication, or, Why Not To Use PGP* [2]. Le but était d'offrir une réponse à deux soucis posés par GPG :

- Le manque de sécurité persistante lié au fait que les clefs privées de GPG sont des clefs de long terme. La confidentialité persistante, appelée *Forward secrecy* en anglais, empêche un attaquant de déchiffrer les échanges précédents en cas de compromission de la clef privée.
- Son authentification forte ne permet pas la répudiabilité, les messages étant signés avec les clefs de long terme. La répudiabilité permet de désavouer le contenu d'une conversation précédente. Comme dans la vie réelle, lorsque deux personnes discutent, c'est la parole de l'une contre celle de l'autre pour savoir ce que chacun a dit ou non.

Le papier était accompagné (chose assez rare pour être mentionnée) d'une implémentation complète dans un logiciel de chat populaire : Gaim, aujourd'hui connu sous le nom de Pidgin. En effet, OTR cible davantage la messagerie instantanée que les échanges asynchrones (de type courriels par exemple), déjà couverts (tant bien que mal) par GPG.

Autre trait inhabituel du papier, son modèle de menace (*threat model* en anglais). En effet, d'habitude, les papiers de cryptographie racontent l'histoire de Bob et d'Alice tentant de communiquer, pendant qu'Eve les espionne de diverses manières. Ici, il s'agit non seulement de ça, mais également du fait qu'Alice ou Bob pourrait décider de trahir son interlocuteur à tout moment.

OTR vise en fait à fournir les mêmes caractéristiques qu'une conversation en tête à tête dans une pièce entre Alice et Bob :

1. Personne ne sait ce qui y est dit, à part Alice et Bob.
2. Personne ne sait ce qui y a été dit, sauf si Alice et Bob le racontent.
3. Personne ne peut modifier la conversation sans qu'Alice et Bob ne le remarquent.
4. Personne ne peut prouver ce qui y a été dit, **pas même Alice et Bob !**

En plus de 10 ans d'existence, OTR a reçu plusieurs *peer-review*, certaines débouchant sur des attaques concrètes, a évolué pour les corriger, pour se simplifier, ajouter des fonctionnalités... Bref, c'est un protocole sérieux, qui a été consciencieusement scruté par des experts en mathématiques, cryptographie, programmation et sécurité informatique. C'est sûrement pour ça qu'il est implémenté dans bon nombre de logiciels de messagerie instantanée : Pidgin, Irssi, WeeChat, Gajim, Adium, TorBirdy, Cryptocat...

2. FONCTIONNEMENT

2.1 Transport

Le protocole OTR est agnostique quant à sa couche de transport : chaque message est envoyé encodé en *base64*, un encodage utilisant uniquement 64 caractères affichables : 26 lettres en minuscule, 26 en majuscule, 10 nombres, ainsi que les signes +, / et =. Cela rend possible l'utilisation d'OTR sur n'importe quel protocole permettant de transmettre du texte. Par exemple, Pidgin va jusqu'à proposer de s'en servir sur le chat Facebook.

Pour initier une conversation OTR, il y a deux méthodes possibles. On peut envoyer soit un *tag* OTR de la forme `?OTR` suivi d'un indicatif de version, soit un ensemble d'espaces et de tabulations. Les deux ont une différence sémantique : le premier indique qu'Alice *aimerait* que Bob ait une conversation OTR avec elle, tandis que la deuxième indique que son client est *capable* de converser en OTR.

2.2 Authentification (SMP)

Qui a déjà tenté d'utiliser GPG sait à quel point la vérification d'empreintes de clefs publiques est une étape fastidieuse, qui a très vite tendance à devenir kafkaïenne quand il s'agit d'établir un canal sécurisé, pour échanger les empreintes, pour les valider, pour s'en servir pour pouvoir communiquer à travers un... canal sécurisé. Bref, ça n'est pas pratique du tout, encore moins quand on essaye de s'en servir avec quelqu'un qui n'est pas habitué à ce genre de manœuvres.

C'est pour ne pas reproduire cette erreur qu'OTR propose une alternative plus facile à utiliser : le SMP, pour *Social Millionaire Protocol*.

L'idée vient du *Problème des millionnaires* posé par Andrew Yao en 1982 [3], dans lequel deux millionnaires veulent savoir lequel d'entre eux possède la plus grosse fortune, sans pour autant en divulguer le montant. La solution repose sur des mathématiques de niveau lycée, qui ne seront pas détaillées dans cet article. Ici, le montant de la fortune sert en fait de secret partagé, connu à la fois d'Alice et de Bob, qui peuvent s'assurer mutuellement qu'ils le connaissent bien, tout en ne le divulguant jamais.

Cette méthode fait partie des méthodes d'authentification dites de type *ZKIP*, pour *Zero Knowledge Identity Proof*, en français : preuve à divulgation nulle de connaissance. En effet, dans les systèmes plus classiques, comme une authentification sur un site web, le serveur demande le mot de passe, et le client lui transmet. Les problèmes arrivent quand l'utilisateur se connecte à un attaquant se faisant passer pour le vrai site, auquel il donnera son mot de passe. Dans une authentification de type ZKIP, l'utilisateur ne divulgue rien que le site (ou l'attaquant) ne connaisse déjà.

Un attaquant passif (qui se contente d'écouter le trafic) n'aurait aucune information quant au mot de passe ; un attaquant actif (qui modifie le trafic) pourrait savoir uniquement s'il a deviné ou non

ledit secret, et faire échouer le protocole, mais rien de plus. Il serait incapable de se faire passer pour l'un des deux protagonistes.

Dans le cadre d'OTR, il est également possible d'utiliser une question (transmise en clair), sa réponse étant la valeur secrète. Par exemple « Quand on s'est rencontré pour la dernière fois, qu'est-ce que je t'ai fait à manger comme dessert ? », avec comme réponse « Un chausson aux pommes ». Ce qui est bien plus convivial et facile à utiliser qu'une vérification réciproque d'empreintes de clefs publiques comme on a l'habitude d'en faire.

2.3 Confidentialité persistante

Traditionnellement (avec GPG), pour envoyer un message à Bob il suffit à Alice de récupérer la clef publique de Bob, de chiffrer le message puis de lui envoyer, afin qu'il soit reçu, déchiffré avec la clef privée, puis finalement lu par Bob.

Le souci se pose quand Eve enregistre les échanges, et parvient à voler la clef privée (saisie, vol, virus...). Elle est alors en mesure de déchiffrer l'intégralité des messages échangés.

Pour répondre à ce problème, les protocoles modernes de cryptographie comme OTR utilisent un système de clefs éphémères : au lieu d'utiliser les clefs de long terme pour chiffrer/déchiffrer le message, elles servent uniquement à authentifier la négociation de clef éphémère, qui a lieu pour OTR au début d'un échange.

Il existe plusieurs méthodes de négociation de clef éphémère, OTR utilisant la plus connue : Diffie-Hellman. Elles permettent la création d'un secret partagé (ici, des clefs de chiffrement *temporaires*) sans jamais le faire transiter. Le fait que ce soit ces clefs éphémères qui soient utilisées pour le chiffrement permet d'empêcher que des échanges passés soient déchiffrables si les clefs de long terme venaient à être compromises, car elles interviennent uniquement pour l'authentification, et pas pour le chiffrement.

2.4 Répudiabilité

Bob et Alice ont une conversation au sujet de leur liaison secrète, mais Bob menace de tout révéler au grand jour ! Heureusement, la propriété de répudiabilité d'OTR signifie que Bob ne peut pas prouver qu'Alice était réellement l'auteure des messages, même s'il en était sûr pendant la conversation.

Ce type de propriété est appelé *répudiabilité faible ;* il existe également de la *répudiabilité forte*, et OTR fournit les deux. La première indique que si l'un des protagonistes rend public un message reçu, il est possible qu'il l'ait entièrement fabriqué.

La seconde garantit que si un message vient à être rendu public, il peut avoir été fabriqué par n'importe qui, et non pas uniquement par Alice ou Bob : chacun d'entre eux pourrait donc légitimement nier que la conversation ait eu lieu.

Le fait qu'OTR soit un protocole authentifié n'est pas incompatible avec ces propriétés de répudiabilité, même si cela peut sembler contre-intuitif au premier abord. L'astuce est de ne pas utiliser de signatures issues de clefs de long terme, comme le fait GPG, mais plutôt des MAC (*Message Authentication Codes*), garantissant l'intégrité d'un message, à l'aide d'une clef éphémère connue à la fois de l'expéditeur et du destinataire.

Pour chaque message à envoyer :

1. l'expéditeur chiffre le message ;
2. calcule son MAC ;

3. envoie le message chiffré ainsi que son MAC associé au destinataire ;
4. destinataire recalcule le MAC pour vérifier que le message n'a pas été altéré, et qu'il provient bien du bon expéditeur ;
5. finalement, le destinataire déchiffre et lit le message.

La beauté de la chose est que pour vérifier qu'un message est authentique, il faut posséder la clef du MAC, ce qui permet donc de **contrefaire** ce message ! En effet, pour contrefaire un message, il suffit de connaître deux choses : sa clef de MAC (qui est rendue publique une fois ledit message reçu), et sa clef de chiffrement. Grâce à la MAC, il est possible de contrefaire de nouveaux échanges à loisir. Par contre, il est impossible de faire de même pour un message futur. Impossible de trahir son interlocuteur en révélant des échanges, car dès qu'on est en mesure de déchiffrer un message, on ne peut pas prouver son authenticité à qui que ce soit : c'est là-dessus que repose la répudiabilité forte d'OTR.

L'implémentation originale sous forme de plugin pour Gaim était accompagnée d'un outil pour contrefaire des messages, afin que n'importe qui puisse le faire.

3. OTR EN PRATIQUE

Cette partie explique plus en détail certaines parties jugées intéressantes de la dernière version d'OTR, la 3.4. Pas d'inquiétude, les primitives cryptographiques utilisées sont expliquées en détail, à la fois dans le cas général (la cryptographie ne se limite pas à OTR), et dans le cas d'OTR.

3.1 Primitives

3.1.1 Hashage et HMAC

Une fonction de hashage est une fonction qui prend en entrée une donnée de taille arbitraire, et retourne une donnée de taille fixe, appelée *hash*, ou *condensat*, voire *empreinte* en français.

Par exemple, l'opérateur modulo est une fonction de hashage :

Fichier

```
1 % 42 = 1
43 % 42 = 1
1337 % 42 = 35
```

Celle-ci projette l'ensemble des entiers dans l'ensemble `[0, 41]`.

Une fonction de hashage dite *cryptographique* doit satisfaire quatre propriétés :

1. Il est facile de calculer le hash de n'importe quelle donnée ;
2. Il est impossible de retrouver un message à partir de son hash ;
3. Il est impossible de modifier un message sans modifier son hash ;
4. Il est impossible de trouver deux messages avec le même hash.

Notre fonction *modulo* ne satisfait ni la troisième, ni la dernière propriété.

Grâce aux fonctions de hashage cryptographique, il est possible de construire ce qu'on appelle un *HMAC*, pour *(key-)Hash Message Authentication Code* (*code d'authentification d'une empreinte cryptographique de message avec clé* en français). Il s'agit d'une primitive permettant de vérifier l'intégrité de données, mais aussi l'authenticité.

3.1.2 Échange de clefs Diffie-Hellman

Un échange de clefs Diffie-Hellman est une méthode par laquelle deux entités se mettent d'accord sur un secret sans qu'un attaquant passif puisse le trouver.

Son fonctionnement repose sur le problème du logarithme discret, qui dit *grosso modo* que dans un ensemble modulo p, pour α et β donnés dans cet ensemble, il est très difficile de trouver k pour $\alpha^k = \beta$.

Une propriété intéressante de la multiplication (une élévation à une puissance étant une suite de multiplications) est que $(a^x \% p)^y = a^{(x \cdot y)} \% p$.

Concrètement, Alice et Bob se mettent d'accord sur deux nombres, g et p, génèrent chacun un nombre aléatoire secret, et procèdent comme suit :

1. Alice envoie $A = g^a \% p$ à Bob, qui l'élève à la puissance b, donnant $K = g^{(a \cdot b)}$.
2. Bob répond par $B = g^b \% p$ à Alice, qui l'élève à la puissance a, donnant $K = g^{(a \cdot b)}$.

Alice et Bob ont donc créé un secret partagé, K.

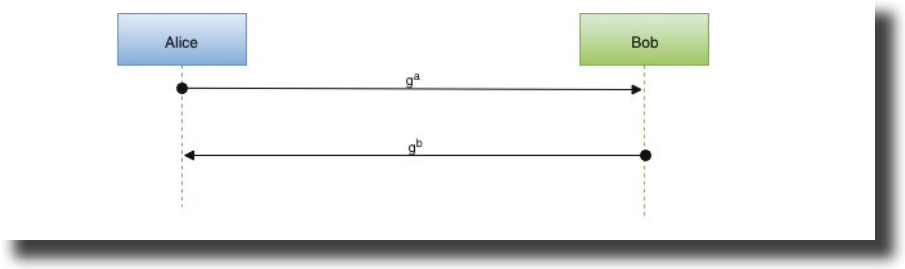


Fig. 1 : Échange Diffie-Hellman entre Alice et Bob.

Mais pour ceux qui n'aiment pas les maths, un échange Diffie-Hellman peut être facilement expliqué avec des pots de peinture. Si on admet que le fait de retrouver deux couleurs à partir de leur seul mélange est impossible, alors : Figure 2.

Ce type d'échange est en général utilisé comme primitive pour construire des protocoles de création de clefs éphémères.

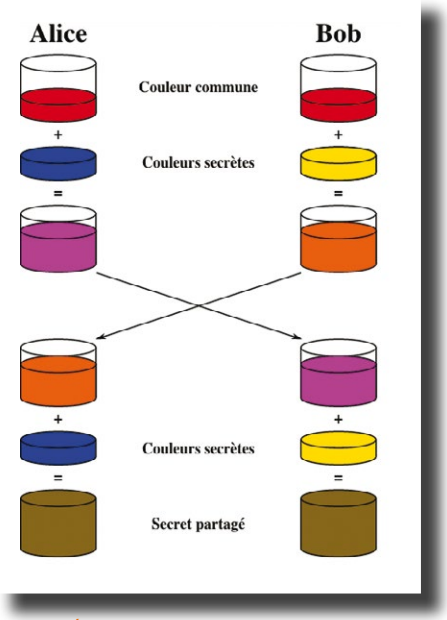


Fig. 2 : Échange Diffie-Hellman avec de la peinture.

3.1.3 Chiffrement/signature

La cryptographie traditionnelle (aussi appelée *cryptographie symétrique*) utilise la même clef pour chiffrer et déchiffrer, ce qui pose une multitude de problèmes, comme celui de la distribution de clefs. Heureusement, aux alentours des années 80, Hellman et Diffie (encore eux !) commencèrent à dessiner une cryptographie asymétrique, avec deux clefs : une clef publique, distribuée à tous, et une clef privée, gardée secrète.

La clef publique permet à tous de *chiffrer* un message, qui ne peut être *déchiffré* que par la clef privée.

La clef privée permet de *signer* un message, signature qui peut être *vérifiée* par tous, grâce à la clef publique.

3.2 Échange de clefs et envoi de message

Maintenant que les primitives ont été définies, entrons dans le vif du sujet. Alice et Bob possèdent chacun une paire de clefs : une clef publique, et une clef privée. Cette paire est appelée *master key*.

3.2.1 Échange de clefs

3.2.1.1 Échange de clefs dans le cadre d'OTR 1.0

Pour s'échanger des messages, ils effectuent un AKE, pour *Authenticated Key Exchange*, un échange de clefs authentifié de type Diffie-Hellman : Alice et Bob signent chacun de leurs échanges Diffie-Hellman avec leur propre *master key*.

Le secret partagé est ensuite hashé pour donner une clef de court terme (aussi appelée clef éphémère), qui est elle-même hashée pour donner la clef de MAC.

Cette étape leur a permis de :

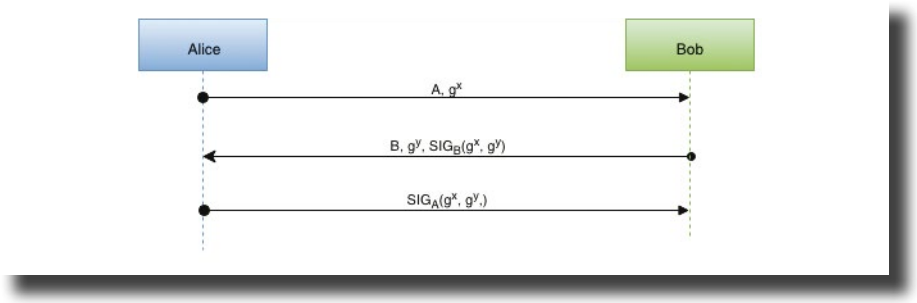


Fig. 3 : Diffie-Hellman authentifié.

1. Créer une clef éphémère, amenant la propriété de confidentialité persistante : si leurs *master keys* viennent à être compromises, il ne sera pas possible de déchiffrer les messages précédents.
2. S'assurer de l'absence d'attaquant actif. En effet, les échanges Diffie-Hellman étant signés avec les *master keys*, il est impossible pour un attaquant de se placer entre Alice et Bob sans qu'ils ne le remarquent.

3.2.1.2 Attaque sur l'échange de clefs d'OTR 1.0

Cette méthode, utilisée dans la première itération du protocole est malheureusement insuffisante. En effet, elle permet à Eve de se placer entre Alice et Bob de manière active :

- Alice envoie g^x a Eve, ainsi que sa signature ;
- Eve envoie g^x à Bob, qu'elle signe avec sa clef ;
- Bob répond donc à Eve g^y, signé avec sa clef à lui ;
- Eve envoie à Alice la réponse de Bob, qui est donc signée avec la clef de Bob.

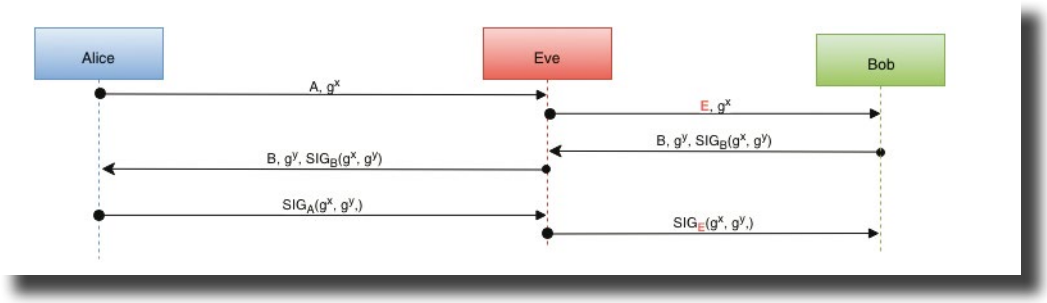


Fig. 4 : Attaque de type misbinding.

Bien qu'incapable de déchiffrer le trafic, Eve a réussi à faire croire à Bob que sa clef de long terme est celle d'Alice. Une fois que Bob et Alice auront effectué une authentification mutuelle avec l'aide du protocole SMP, Bob pensera qu'Eve est Alice. Ce qui permettra à Eve de se faire passer pour Alice durant les prochaines conversations !

La solution évidente (utilisée par ISO-9796) serait d'inclure la clef publique du correspondant dans les échanges Diffie-Hellman, mais une telle manœuvre offrirait une preuve qu'Alice et Bob ont bien conversé (la clef publique de Bob serait dans un message signé par Alice, et inversement), compromettant la propriété de répudiabilité forte d'OTR.

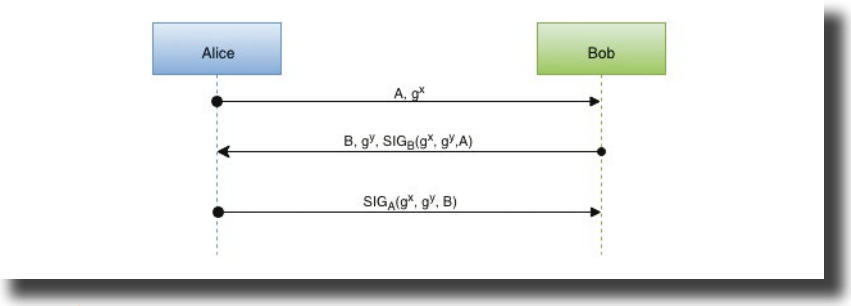


Fig. 5 : Échange de clefs de type ISO-9796.

Un autre type d'attaque est également possible : une attaque de type *replay*. Dans un protocole d'échange de clefs, la seule manière de se faire passer pour quelqu'un dans l'échange actuel est de compromettre la *master key* d'un protagoniste. En d'autres termes, la compromission d'échanges précédents ne devrait pas affecter des échanges futurs. Or, si un attaquant arrive à trouver une valeur x' utilisée par Alice durant une session précédente, il lui suffira d'envoyer $g^{x'}$ et la signature associée, pour établir une session avec Bob, en se faisant passer pour Alice !

Ces failles furent publiées par Mario Di Raimondo, Rosario Gennaro et Hugo Krawczyk, en 2005, dans leur papier intitulé *Secure Off-the-Record Messaging* [4].

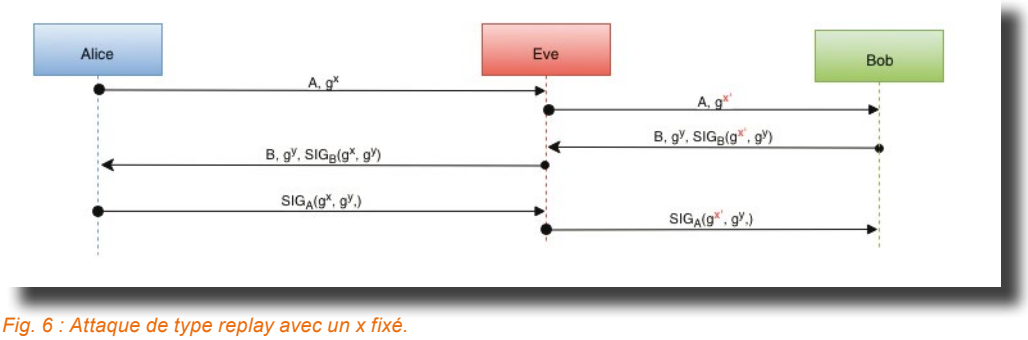


Fig. 6 : Attaque de type replay avec un x fixé.

3.2.1.3 Échange de clefs dans le cadre d'OTR 2.0

Pour corriger les vulnérabilités évoquées plus haut, la version 2.0 d'OTR change d'AKE : elle utilise le protocole SIGMA (pour *SIGn-and-MAc*), développé en 1995, par les développeurs d'IPSEC pour remplacer leur ancien protocole d'échange de clefs **[5]**. L'idée de SIGMA est de lier le secret partagé aux *master keys* des interlocuteurs, avec une signature et un MAC, d'où son nom. Pour ce faire, il établit un canal chiffré, dans lequel se déroule un échange Diffie-Hellman classique (donc non authentifié), grâce auquel l'authentification mutuelle se fera : bref, un gros sandwich de cryptographie en trois couches, la dernière servant à vérifier la première.

Une telle stratégie permet d'empêcher Eve de se faire passer pour Alice et d'effectuer des attaques par replay. Un effet de bord intéressant de la transposition de l'authentification à l'intérieur d'un canal confidentiel est la dissimulation des clefs publiques d'Alice et de Bob à un attaquant passif, augmentant un peu plus l'anonymat de la conversation.

Le protocole SIGMA étant fastidieux (mais simple), il ne sera pas détaillé ici, mais la lecture du papier original est vivement recommandée aux curieux : **http://webee.technion.ac.il/~hugo/sigma.html**.

3.3 Envoi de message, et régénération de clefs

Ensuite, le message à proprement parler est envoyé d'Alice à Bob. Il est chiffré à l'aide d'AES-CTR (un chiffrement symétrique) grâce à la clef éphémère précédemment générée, afin de garantir sa confidentialité, et son MAC est envoyé en clair avec lui, calculé avec HMAC-SHA1 et sa clef de MAC.

Pour chaque message envoyé, Alice et Bob refont un échange Diffie-Hellman, afin de générer une nouvelle clef éphémère et une clef de MAC. Chaque échange est authentifié grâce à la clef de MAC précédente. Une fois l'échange terminé, l'ancienne clef de MAC est publiée.

En revanche, les clefs de chiffrement ne le sont pas !

Le fait d'utiliser la clef de MAC actuelle pour garantir l'intégrité de la suivante permet de se prémunir contre les réordonnancements/la suppression de messages : par exemple, en envoyant un mot par message, il y a une différence notable entre « Je suis content » et « Je [ne] suis [pas] content ».

3.3.1 Limites

Dans cette partie, le terme *adversaire* désigne l'entité (agence de renseignement, compagnon, force de police, parent...) tentant de briser la confidentialité de la conversation, avec l'aide, volontaire ou contrainte de Bob, pour confondre Alice.

Si Bob était malveillant dès le début de la conversation, il lui suffirait de donner sa *master key* à l'adversaire, qui aurait la preuve que la conversation n'est pas contrefaite, car elle aurait lieu directement entre lui et Alice.

La répudiabilité forte ne protège pas non plus d'un adversaire capable d'enregistrer l'intégralité de la conversation, car il serait impossible de prétendre qu'un message a été contrefait. Ce problème peut être résolu en utilisant des réseaux anonymisants, comme Tor par exemple. Il serait bien sûr possible pour Bob d'enregistrer le trafic, mais s'il n'était pas malveillant dès le départ, il est raisonnable de penser que l'adversaire ne lui fera pas confiance pour lui donner une capture réseau de l'intégralité de la conversation sur sa seule bonne foi.

3.4 Utiliser OTR

Comme évoqué au début de l'article, grâce à son encodage en base64, OTR est implémenté dans une multitude de clients, pour à peu près n'importe quel protocole. Par exemple, sur Debian, pour utiliser OTR avec Pidgin, il suffit de taper `apt install pidgin-otr`, pour Irssi `apt install irssi-plugin-otr`. D'autres applications le proposent nativement, comme Jitsi (un client de VOIP en Java), TorBirdy (le client de messagerie instantanée du projet Tor), Cryptocat (une application web), TextSecure (un client mobile par le Guardian Project)...

Par exemple, avec Pidgin : Figure 7, 8 et 9, page suivante.

Bref, OTR est dans plein de clients, et est facile à utiliser.

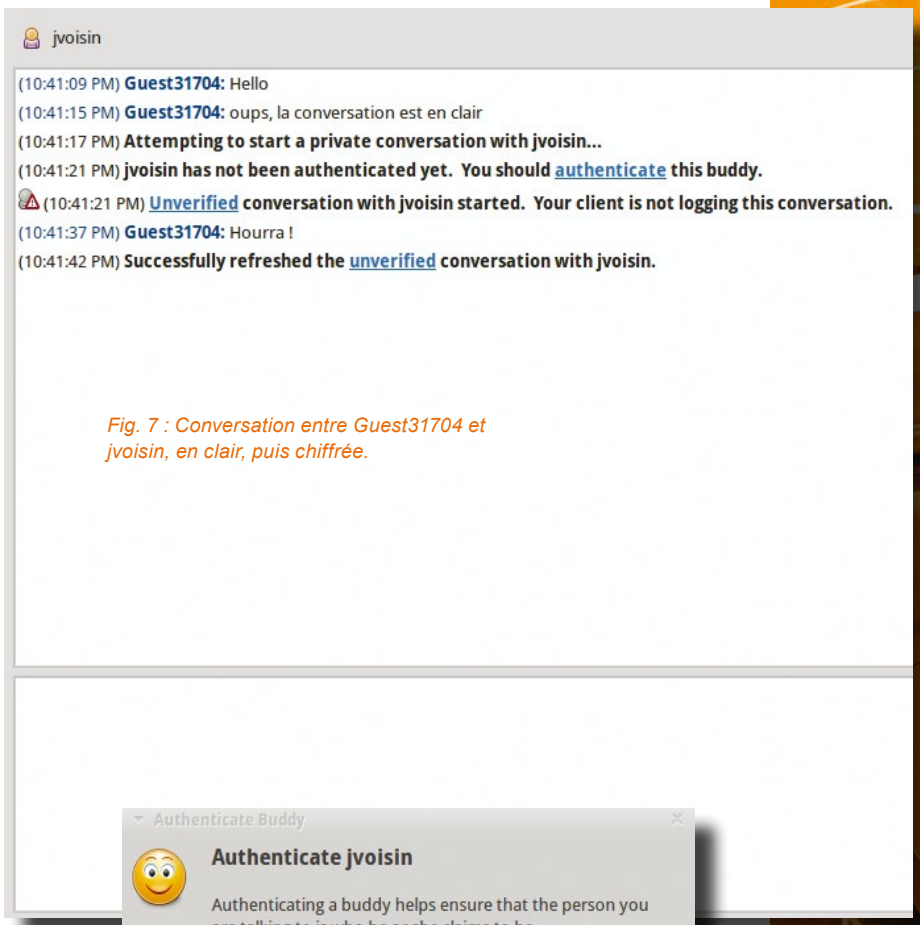


Fig. 7 : Conversation entre Guest31704 et jvoisin, en clair, puis chiffrée.

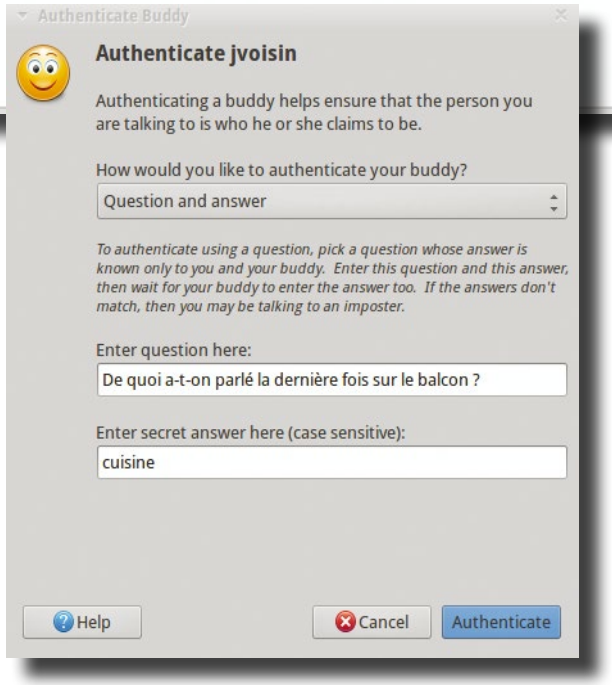


Fig. 8 : Dialogue d'authentification de type question/réponse.

4. HÉRITAGES, ÉCUEILS, ET DÉVELOPPEMENTS FUTURS

En tant que pionnier dans le domaine du chiffrement de messagerie instantanée et de la répudiation, OTR a engendré un héritage important.

Aujourd'hui, l'implémentation de référence d'OTR, libotr, n'est plus activement développée. Elle est en phase de maintenance : si des bugs apparaissent, ils sont corrigés, mais aucune nouvelle fonctionnalité n'a été ajoutée depuis des années.

D'autres implémentations, comme python-otr, ocaml-otr, otr4j... continuent de fleurir, mais le protocole n'a plus bougé depuis sa version 3, qui commence à se faire vieille.

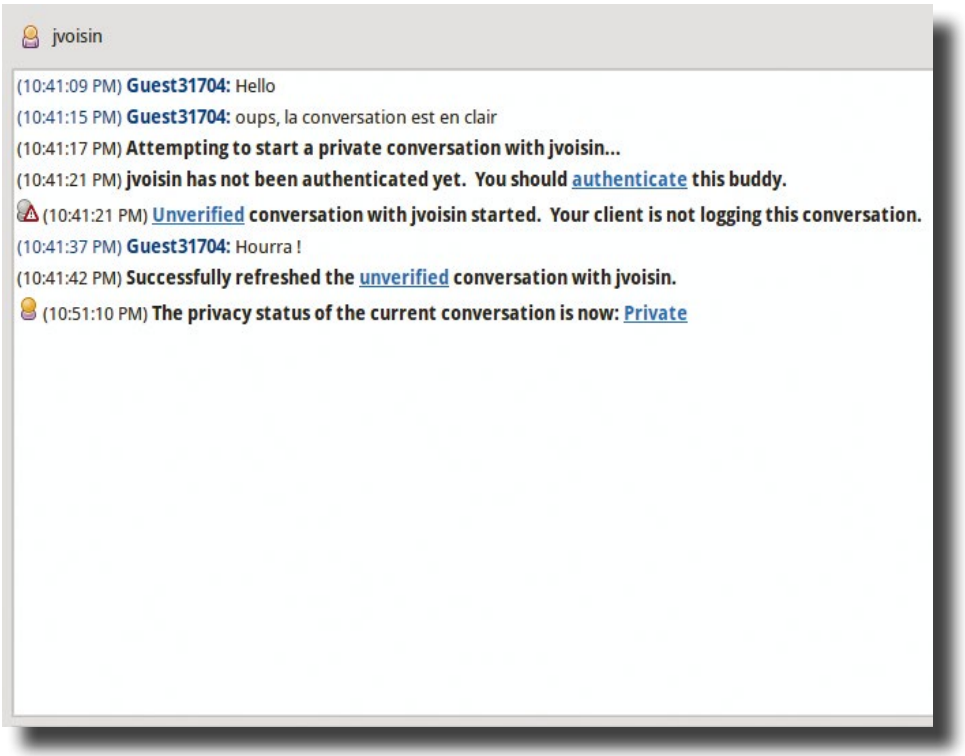


Fig. 9 : Une fois que Guest31704 a correctement répondu à la question secrète, les échanges sont chiffrés et authentifiés.

Par exemple, la publication des MAC ne permet qu'à ceux ayant enregistré la conversation de contrefaire de nouveaux messages : la répudiation serait encore plus forte si n'importe qui pouvait le faire.

L'avènement des communications mobiles amène de nouvelles problématiques, principalement en raison de leur nature asynchrone : les utilisateurs ne sont plus forcément connectés en même temps. Mais qui dit communications mobiles dit aussi soucis de connectivité : les AKE en trois échanges (comme ceux utilisés par OTR) tendent à être remplacés par des AKE en deux échanges seulement, pour gagner du temps et de la bande passante.

Un autre reproche souvent émis à l'encontre d'OTR est le fait qu'il ne permette qu'à deux personnes à la fois de converser en toute intimité. Un papier nommé *Improved Group Off-the-Record Messaging* **[6]** publié en 2013 liste les principaux obstacles à surmonter avant de pouvoir créer un protocole avec les mêmes garanties qu'OTR. Mais des initiatives existent, comme le projet *np1sec* **[7]**, crée pour le logiciel *Cryptocat*, qui est au stade de l'implémentation (le projet cherche à embaucher un développeur C++ d'ailleurs), ou encore mpOTR **[8]**, qui lui n'en est qu'à l'ébauche.

Bien que des initiatives existent (mpOTR, Axolotl, np1sec...), aucune ne s'est encore dégagée de la masse.

Néanmoins, c'est par ses primitives cryptographiques qu'OTR accuse le plus son âge. En 2004, quand OTR a été créé, l'algorithme DSA était un bon algorithme de signature cryptographique. Malheureusement, il n'est pas évident à implémenter correctement (Sony en a fait les frais : le piratage de sa console PS3 a pu avoir lieu grâce à une mauvaise implémentation de DSA.) Son algorithme de hashage, SHA-1 commence

également à montrer des signes de faiblesse, et n'est plus utilisé dans le cadre des certificats TLS, car considéré comme trop faible. Enfin, la taille du modulo Diffie-Hellman (1536 bits) est inférieure aux recommandations actuelles, qui conseillent d'utiliser un modulo d'au moins 2048 bits.

Heureusement, la relève est là : Signal. Une application mobile écrite par Open Whisper Systems, la société de Moxie Marlinspike. Anciennement connue sous le nom de TextSecure, elle utilise une version plus compacte et moderne d'OTR **[9]**, qui corrige les soucis énumérés plus haut. L'application est (évidemment) open source, disponible sur Android et iPhone, et bientôt sous forme d'un client lourd.

CONCLUSION

Bien que vieillissant et possédant des défauts, OTR est toujours le protocole de référence pour assurer confidentialité et répudiation lors d'échanges instantanés. L'avancement en matière de cryptographie et de puissance de calcul le mettent peu à peu hors course, mais il a encore quelques années devant lui avant d'être remplacé, ou qui sait, peut-être mis à jour dans une quatrième version. ■

RÉFÉRENCES

[1] Laura Poitras, Citizenfour : **https://citizenfourfilm.com/**

[2] **https://otr.cypherpunks.ca/otr-wpes.pdf**

[3] Andrew C. Yao, « Protocols for secure computations », FOCS, 1982, 2013 IEEE 54th Annual Symposium on Foundations of Computer Science, 2013 IEEE 54th Annual Symposium on Foundations of Computer Science 1982, pp. 160-164, doi:10.1109/SFCS.1982.88

[4] Mario Di Raimondo, Rosario Gennaro et Hugo Krawczyk, Secure Off-the-Record Messaging, 2004 : **https://www.dmi.unict.it/diraimondo/web/wp-content/uploads/papers/otr.pdf**

[5] Hugo Krawczyk, 2003, SIGMA: The « SIGn-and-MAc » Approach to Authenticated Diffie-Hellman and Its Use in the IKE-Protocols : **http://www.iacr.org/cryptodb/archive/2003/CRYPTO/1495/1495.pdf**

[6] Liu, Hong; Vasserman, Eugene Y. ; Hopper, Nicholas, Improved Group Off-the-Record Messaging, 2013 : **http://krex.k-state.edu/dspace/handle/2097/20141**

[7] **https://learn.equalit.ie/wiki/Np1sec**

[8] **https://cypherpunks.ca/~iang/pubs/mpotr.pdf**

[9] **https://github.com/trevp/axolotl/wiki**

DES CRYPTOPARTIES AUX SALONS VIE PRIVÉE

Raphaël VINOT

Pour le grand public, « crypto » est un mot inquiétant : il semble impliquer des formules compliquées et beaucoup de connaissances avant de pouvoir envisager de participer à un événement ayant ce mot dans le titre. Cet article est un retour d'expériences suite à presque deux ans en tant que coorganisateur de salons vie privée au Luxembourg.

Les premières cryptoparties auxquelles j'ai participé étaient organisées par des geeks, pour des geeks et on parle d'hypothétiques attaques gouvernementales. Ça nous amuse, ça fait peur aux nouveaux, ça casse pas 3 pattes à un canard et après quelques itérations, on se rend compte que l'on parle toujours de la même chose, avec les mêmes gens et que l'on avance pas trop.

1. L'ORDINATEUR, CET OUTIL

La raison principale de cet état de fait est la suivante : les non-geeks utilisent un ordinateur pour faire quelque chose, en général, travailler ou s'amuser. Et ils ne s'amusent pas en utilisant un ordinateur : c'est un outil, comme une voiture. Il est possible d'être passionné, mais en définitive, on veut généralement simplement utiliser la voiture pour aller faire les courses, et pas la démonter tous les matins pour changer les pneus.

En discutant avec d'autres organisateurs, et des gens qui bien qu'intéressés par le sujet n'osaient pas venir, il est devenu très clair que ce n'est pas en attachant « party » à crypto que l'on allait les rassurer. Et ce sans dire que « party » est tout relatif : faire la fête en parlant de problèmes de sécurité n'est pas nécessairement compatible avec la définition de fête du commun des mortels.

C'est la raison principale qui nous a fait changer vers un nom plus inclusif : un des premiers termes que j'ai entendu est chiffrofête qui a le mérite d'être amusant, mais personne extérieur à la communauté ne peut savoir ce que ça veut dire. C'est pourquoi le nom qui s'est imposé dans ces derniers mois est salon vie privée (*privacy salon* en anglais) qui a pour avantage principal de ne pas contenir crypto, mais aussi d'ouvrir le débat sur d'autres problématiques que la cryptographie : la majorité d'entre nous n'a pas vraiment d'adversaires en capacité d'utiliser de manière ciblée des techniques avancées et coûteuses.

Il va sans dire que les gouvernements ont de plus en plus de capacités d'interception et d'exploitation inquiétantes ciblant tout le monde de manière indiscriminée, mais ces capacités ne sont pas tellement différentes de celles des entreprises privées dont le travail est d'analyser notre utilisation d'Internet au quotidien.

Par contre, il est devenu très clair que notre vie est connectée en permanence et que nous laissons des traces partout, volontairement ou involontairement, mais aussi que la majorité des utilisateurs d'Internet ne sait pas vraiment comment toute cette infrastructure fonctionne. Avant même de parler de cryptographie en tant que telle, il faut déjà expliquer ce qu'est le cloud, en quoi HTTPS consiste, comment le réseau fonctionne, qu'il y a des choses physiques derrière les vidéos de chats que l'on regarde sur YouTube, et seulement ensuite parler outils de chiffrement, modèles de menace et autres maliciels gouvernementaux.

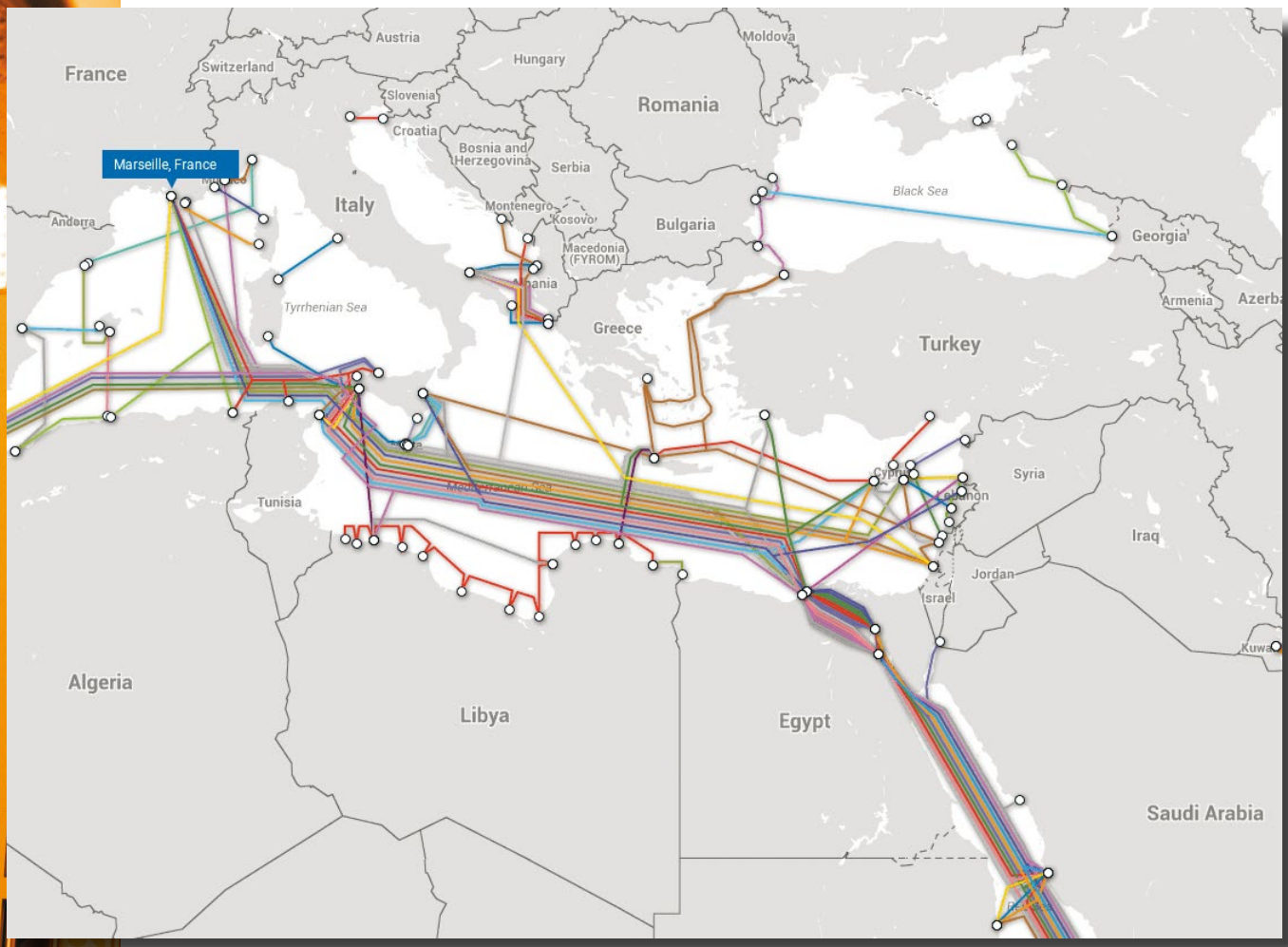
Notre but ultime est toujours de parler cryptographie, mais pour être efficace, il faut susciter l'intérêt dans un contexte qui parle à un public curieux, mais pas passionné.

2. LES CÂBLES, LES ACTEURS ET LES DIFFÉRENTES JURIDICTIONS

Quel que soit le service que vous utilisez sur Internet, vous allez toujours utiliser l'infrastructure physique mise en place par plus de 250 compagnies **[1]**. Si l'on s'intéresse aux acteurs locaux pour le RIPE (Europe, Moyen-Orient, Asie centrale), on arrive à plus de 13000 **[11]**. Toutes ces organisations, majoritairement privées, sont installées dans beaucoup de pays différents,

se conforment à des règles différentes, sont plus ou moins proches des gouvernements, et les gouvernements ont plus ou moins de poids sur leur fonctionnement au quotidien, comme en Égypte en 2011 [2] par exemple.Il est pratiquement impossible de savoir à tout instant le chemin que vos données vont emprunter. Rien de mal à ça, c'est la façon dont le réseau internet a été créé et c'est une des raisons pour lesquelles il est si efficace et fiable.

Néanmoins, il existe des points de transits plus conséquents que d'autres. Pour prendre un exemple français, la probabilité qu'un internaute vivant au Maghreb ou dans la péninsule arabique voit son trafic passer par Marseille à un moment ou à un autre va être élevée.



Un autre cas notable va être Bude, dans les Cornouailles au Royaume-Uni qui est connecté à 6 fibres majeures. Quelques-unes allant vers l'Europe et les USA, mais aussi vers l'Asie et vers le golfe de Guinée. Bude est aussi bien connue pour héberger une station d'écoute du GCHQ [3], et les documents Snowden contenaient des informations expliquant que la NSA a utilisé ce site pour des tests d'interception sur la fibre optique [4].

Comme on peut le voir, les attaques sur l'infrastructure existent et des gouvernements les mettent en place de manière industrielle. Elles sont relativement simples à condition d'avoir un accès physique aux infrastructures, ce qui implique généralement des capacités gouvernementales. Je ne suis pas au courant d'attaques provenant de compagnies directement, mais je peux me tromper.

La question que vous vous posez sûrement à ce moment est : que faire ? S'il est possible d'intercepter le trafic trivialement, nous sommes coincés, non ?

Dans la pratique, il est peu probable qu'un particulier soit directement ciblé personnellement par ce genre d'activités.

En admettant que ce soit le cas, les outils utilisés pour dupliquer le trafic vont capturer une masse très importante de données. Il est très important de savoir que si le trafic en clair est simple à analyser, autant le trafic chiffré est d'une autre complexité.C'est pourquoi la question de faire confiance ou non au réseau devient accessoire. S'il y a une raison à retenir pour tout chiffrer, tout le temps, c'est bien celle-là : à partir du moment où une communication est chiffrée, il est possible d'utiliser tout réseau à sa disposition sans vraiment hésiter. Comme se connecter à Internet lors d'une conférence, ou dans un hôtel.

Pour un utilisateur lambda qui utilise du chiffrement, la question d'avoir confiance dans le réseau n'est pas pertinente : rien ne devrait jamais passer en clair de toute façon donc il n'y a rien que le réseau, et n'importe quel acteur ayant un accès physique dessus, puisse faire sans nécessiter beaucoup de ressources. Oui, toi, le nerd dans le fond, qui trépigne sur ta chaise, nous allons parler des métadonnées dans un instant.

Ça, c'est la bonne nouvelle, vous pouvez vous connecter à Facebook sans grands risques depuis un réseau public. À une condition de taille néanmoins : ne jamais ignorer une alerte dans le navigateur.

La mauvaise nouvelle, par contre, c'est que le chiffrement n'est pas un outil magique qui vous protège de tout. Le réseau a besoin de savoir comment et où envoyer la photo de chat que vous êtes en train d'envoyer sur YouTube, et pour ce faire, il va avoir besoin de métadonnées, qui sont en clair et visibles par toute personne écoutant sur la ligne. Dans le cas mentionné précédemment, il est aisé de savoir que vous vous connectez sur YouTube à une certaine heure, pendant un certain temps, et que vous envoyez une vidéo.

Le chiffrement est uniquement appliqué au contenu de vos conversations. Dans le cas de Facebook, le chiffrement est appliqué sur votre mot de passe, les pages que vous visitez, les personnes avec lesquelles vous parlez, ce que vous leurs dites, mais il est toujours possible de voir que vous vous connectez à Facebook, à quelle heure, depuis où, la masse de données que vous échangez...

Pour le trafic internet c'est encore relativement léger, mais si vous prenez le cas des métadonnées d'un appel téléphonique, ou d'un SMS, c'est beaucoup plus inquiétant (sachez qu'en plus de tout ça, le contenu de la conversation n'est pour ainsi dire pas chiffré) : en plus de ce que je viens de mentionner, vous aurez aussi le numéro de téléphone de votre interlocuteur, vos coordonnées GPS et le temps de communication.

L'agrégation de ces données va fournir des informations très précises concernant votre vie privée.

3. QUI A INTÉRÊT À ACCÉDER À VOS DONNÉES ?

La presse vous parle beaucoup de la NSA, Snowden et les gouvernements, mais dans la pratique, ils ne vont très probablement pas s'intéresser à vous personnellement. Et si vous pensez que c'est le cas, allez au plus vite voir un hackerspace près de chez vous, ils pourront vous donner des recommandations.

Ce qu'un gouvernement va en général faire, c'est capturer un maximum de choses (fonction de ses capacités), et ensuite faire le tri. Tout ce qui est en clair est extrêmement rapidement analysé/analysable, mais le trafic chiffré va demander plus de travail.

Pour ce qui est des compagnies, comme Facebook dont nous parlions plus tôt, le chiffrement est un outil marketing bien plus que de protection pour leurs utilisateurs : si n'importe qui pouvait simplement accéder aux données de leurs utilisateurs gratuitement et proposer de la publicité par exemple, ça irait contre leur business model.

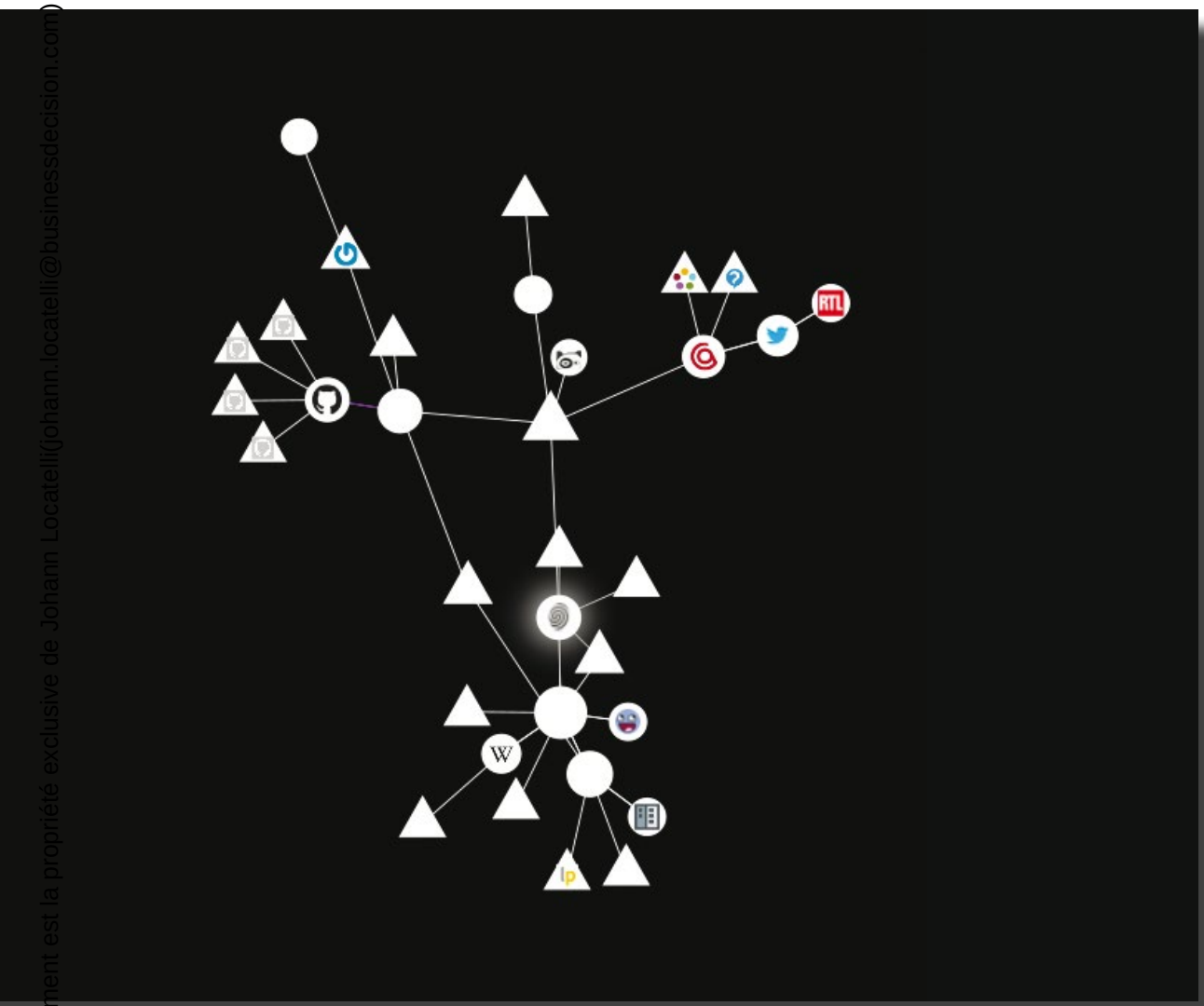
Le risque principal que vous rencontrez au quotidien est de très loin le tracking par des entreprises commerciales voulant vous vendre des services et qui vont pour ça mettre toutes sortes de techniques en œuvre pour collecter un maximum d'informations et construire un profil de vous aussi fidèle que possible. Toutes ces données entrent dans un marché et vont être revendues entre différentes entreprises, le plus souvent sans votre consentement éclairé : vous y avez généralement consenti en acceptant le contrat d'utilisation.

L'outil visuel le plus efficace pour visualiser le tracking auquel vous êtes soumis en simplement surfant sur internet est Lightbeam [5] qui ne fonctionne que sous Firefox, mais a le gros avantage de vous montrer avec précision quels sites ont quels trackers et leur nombre.

Pas si problématique me direz-vous, tant que vous n'êtes pas identifiés sur le site que vous visitez, les trackers ne peuvent pas vous identifier... Dans la pratique, il existe beaucoup de moyens pour réaliser une empreinte de votre navigateur jusqu'à vous identifier de manière (pratiquement) unique : votre adresse IP, la configuration de votre navigateur, les greffons que vous utilisez, les requêtes que vous autorisez, ou bloquez, les polices de caractères... Le meilleur moyen de se rendre compte de ce fait est de visiter Browser Leaks [6].

Ces techniques sont utilisées par toutes les agences de publicité pour créer des profils fantômes des visiteurs, et en se basant sur ces profils, leur proposer des publicités ciblées. Les identifier nommément est totalement superflu à ce point.

L'autre menace principale que vous allez rencontrer viendra des autres personnes qui ont accès à vos comptes (via votre ordinateur ou via vos comptes partagés avec des tiers) : le risque le plus fréquent provient de personnes que vous connaissez : l'attaquant inconnu qui réussit à obtenir un accès à vos données existe, mais il va arriver loin derrière la curiosité déplacée de vos proches ou amis. Un des risques principaux pour un adolescent viendra des parents trop curieux espérant contrôler sa vie discrètement. Dans le cas des adultes, il pourra s'agir d'un compagnon jaloux, ou d'un employeur voulant surveiller ses employés.



Dans les deux cas, les mêmes outils seront utilisés, se comportent comme des malwares, sont extrêmement invasifs et le plus souvent envoient les données collectées quelque part dans le cloud, sans sécurisation effective.

Dans tous ces cas-là, le chiffrement est totalement ineffectif : d'une part l'utilisateur a donné ses données de son plein gré, d'autre part les données sont volées en clair.

Pour donner un exemple, Mspy est un outil créé par une société ayant pignon sur rue et qui ne se cache pas de permettre de surveiller qui bon vous semble (pour peu que vous achetiez l'outil), et ils stockent toutes les données volées dans le cloud. Que se passe-t-il quand ils s'avèrent vulnérables ? Tout le monde peut accéder à des informations très personnelles **[7]**, ce qui a provoqué des campagnes de chantage contre les victimes. Toute société est susceptible d'être vulnérable à un moment ou à un autre, sans exception.

Une autre chose importante à savoir alors que l'on parle des téléphones mobiles : un iPhone jailbreaké pour installer des applications hors du Apple Store est beaucoup plus vulnérable à l'installation de malwares. Du côté d'Android, votre téléphone, s'il est maintenu à jour, sera moins vulnérable non-rooté et avec uniquement des installations venant du Google Store (gardez néanmoins un œil sur les permissions...).

Comme on peut le voir, le chiffrement n'est donc pas vraiment la solution à tous les problèmes, loin s'en faut. C'est un outil parmi d'autres, dont il faut connaître les points forts et les points faibles.

4. CAS PRATIQUES ET OUTILS ASSOCIÉS

Maintenant que nous sommes passés à travers tous ces risques, que faire pour les réduire ? Je parle bien de réduire les risques et de les comprendre, pas les éliminer complètement, parce que c'est impossible.

La bonne nouvelle, c'est qu'il y a des outils simples qui vont vous simplifier la vie, et nous allons les aborder au cas par cas dans les paragraphes suivants, en fonction des risques à prendre en compte.

Le premier et de très loin le plus simple à résoudre pour l'immense majorité des cas que vous allez rencontrer, est de faire en sorte que vous vous connectiez de manière sécurisée (HTTPS) aux sites que vous visitez. Ce greffon s'appelle HTTPS Everywhere **[8]** et va automatiquement rediriger votre navigateur vers la version chiffrée si possible, sans que vous ayez à y penser.

Un autre problème qui se pose de plus en plus fréquemment est la compromission de sites fournissant des publicités à des sites tiers : ils sont une cible de choix pour les attaquants parce qu'ils permettront d'attaquer un grand nombre d'utilisateurs sans dépenser tellement de ressources. Une solution à cela est la mise en place d'ad-blocker comme ublock origin **[9]**. L'autre avantage de ce genre d'outils est qu'ils vont aussi éviter de voir votre navigation être suivie, analysée et vendue à des tiers.

Un autre moyen de se protéger de la revente d'informations personnelles à des tiers sera d'utiliser de fausses informations : ce n'est pas parce que les contrats d'utilisation vous imposent de fournir de vraies informations quand vous utilisez des services en ligne que vous devez obtempérer sans réfléchir.

Notez bien néanmoins qu'utiliser une fausse identité, si découverte, peut mener à la fermeture de votre compte, mais c'est à peu près tout ce que vous risquez dans la pratique. Et combien de comptes à usage unique avez-vous créés dans les derniers jours ? Probablement beaucoup trop.

Assez parlé de vous protéger d'inconnus ou d'organisations, vous partagez sûrement un ordinateur avec des tiers (famille, partenaires, amis, parents, enfants...), auxquels vous allez apporter une confiance variable dans le temps.

Dans ce cas-là, il faut penser à utiliser la navigation privée qui a pour avantage principal de ne pas conserver d'historique de navigation sur l'ordinateur, sans devoir y penser et supprimer des sessions spécifiques. Dans le cas où vous voulez quand même conserver ces données, pensez à utiliser un profil différent par utilisateur : c'est loin d'être sûr, mais tant que les autres utilisateurs ne sont pas trop curieux, ils ne vont pas tomber sur vos données privées par mégarde.

En plus de ça, si vous suspectez que le mot de passe de vos services (Gmail, Facebook, Twitter...) puisse être utilisé par d'autres, mais que vous êtes certains que votre téléphone est toujours avec vous et que personne ne va pouvoir l'utiliser sans votre connaissance, activez l'authentification deux facteurs. De cette manière vous allez bloquer effectivement toute personne essayant d'accéder à vos comptes, même si vous avez partagé votre mot de passe. Pensez néanmoins à vous déconnecter de tous les sites que vous avez utilisé.

Pour avoir une couche de protection supplémentaire, ou si vous utilisez fréquemment un ordinateur partagé, envisagez d'utiliser Tails **[10]** : il s'agit d'un CD (ou d'une clef USB) spécialement créé pour pouvoir démarrer un ordinateur sans y laisser la moindre trace après l'avoir utilisé. De cette manière, personne utilisant le même ordinateur que vous ne pourra accéder à des informations personnelles que vous aurez oublié d'effacer. En plus de cela, Tails va automatiquement utiliser Tor **[12]** pour toutes vos connexions. Je ne vais pas m'étaler dessus, mais Tor peut s'apparenter à un VPN du pauvre : vous allez automatiquement rendre impossible au fournisseur d'accès internet que vous utilisez, ou à l'hôtel de vos vacances, de savoir les sites que vous visitez.

Dans le cadre de l'utilisation de l'ordinateur d'un tiers, ou appartenant à votre employeur, il ne sera pas rare de voir des certificats TLS racines être ajoutés dans les navigateurs. Ces derniers vont lui permettre de déchiffrer toutes les communications que vous pensiez chiffrer. Et ce notamment vers les services que vous allez typiquement visiter sur votre temps libre (Gmail, Twitter...). Si c'est le cas, vos mails personnels, même hébergés en dehors de votre lieu de travail, lui sont visibles.

Un autre point à ne pas négliger dans la protection de vos comptes sur différents sites : les questions secrètes ou tout autres moyens de récupérer l'accès à un compte dont vous allez avoir oublié le mot de passe. Si la réponse à la question est trouvable sur Internet ou que des gens autres que vous peuvent la déduire, la capacité d'un attaquant potentiel à compromette votre compte va y être directement liée, et ce quelle que soit la complexité de votre mot de passe original.

Et pour finir un dernier cas de plus en plus fréquent ces temps-ci : les malwares qui chiffrent l'intégralité de vos données, et demandent une rançon pour vous donner la clef de déchiffrement. Dans l'immense majorité des cas, il est impossible de récupérer les données sans cette clef. La seule solution à ce risque est de faire des sauvegardes, et de les conserver sur des médias hors-ligne (qui ne sont pas connectés à l'ordinateur à protéger), sous peine de prendre le risque de les voir chiffrées aussi.

5. « JE VEUX ÊTRE ANONYME SUR INTERNET, TELL ME EVERYTHING »

C'est une question qui va être posée à chaque événement où l'on parle protection de la vie privée sur Internet. Loin d'être stupide, cette question montre en fait un manque de compréhension de ce qu'est le réseau Internet.

Tout d'abord, de quoi parle-t-on ? D'anonymat (personne ne sait qui vous êtes), ou de pseudonymat (vous êtes identifiés par un pseudonyme) ?

Si l'on parle d'anonymat en général, quel que soit le chiffrement utilisé, le seul moyen d'être anonyme tout le temps pour tout le monde est de n'avoir aucun ami, changer de lieu, d'identité et de matériel tout le temps, de ne parler à personne... et d'éviter Internet en général.

Utiliser un pseudonyme par contre va signifier que vous ne voulez pas donner votre nom usuel pour toutes sortes de bonnes raisons qui vont de ne pas vouloir être importunée sur Internet parce que vous êtes une femme, éviter de connecter des idées politiques à son identité jusqu'à être un lanceur d'alerte voulant se protéger de potentielles répercussions.

Un exemple très efficace que j'ai pu expérimenter plusieurs fois est de voir le pseudonymat comme un moyen raisonnablement sûr pour un ou une adolescente d'éviter de polluer son image sur Internet avec des positionnements personnels dont il ou elle ne sera pas nécessairement fier après quelques années.

La première question à se poser est de savoir qui, ou quoi, ne doit pas connaître votre identité ? Le fournisseur d'accès internet ? Votre interlocuteur ? Le service utilisé pour communiquer ? Toutes ces questions vont impacter les outils et méthodes que vous allez devoir mettre en place pour arriver à vos fins.

Dans le cas du fournisseur d'accès internet, si vous utilisez une connexion qui n'est pas à votre nom et que la connexion au site que vous utilisez est chiffrée, il n'aura pas de moyen simple de savoir qui vous êtes. Pour ajouter une couche de protection et aussi cacher au FAI les sites que vous utiliserez, vous utiliserez un VPN ou Tor.

Si le risque vient de votre interlocuteur, vous allez préférer un service dont vous êtes sûr qu'il ne laisse pas fuiter d'information permettant de vous identifier (nom, IP source...).

Si le service que vous utilisez ne doit pas pouvoir vous identifier, vous allez préférer un compte jetable sur la plateforme et vraisemblablement un chiffrement de bout en bout avec votre interlocuteur de manière à éviter de laisser fuiter le contenu au service.Le dernier point à ne pas oublier est le temps d'utilisation d'une fausse identité : plus elle va être utilisée pendant une longue période, plus le risque de voir votre identité usuelle révélée va être élevé.

D'une manière générale, se noyer dans la masse est le meilleur moyen : il est raisonnable de partir du principe qu'aucun gouvernement n'aura la capacité de casser toutes les sessions TLS à moyen terme. Ce qui nous amène à la question suivante : combien de temps mes données chiffrées doivent rester inviolables ? Tout est cassable, potentiellement toute implémentation peut avoir une vulnérabilité critique et dans tous les cas, même en force brute, toute session chiffrée du passé pourra être cassée à terme si elle est conservée suffisamment longtemps.

Une donnée qui ne doit jamais être publique ne doit pas être transférée, et n'aurait jamais dû être numérisée dans un premier temps.

CONCLUSION

Tout programme informatique a des bugs, ces bugs vont potentiellement être exploités par des attaquants dans un but malveillant. Mais dans la plupart des cas, des données privées vont se retrouver sur Internet parce qu'un développeur a fait une erreur, même si ces données ont été chiffrées quand vous les avez partagées avec un petit nombre d'amis de confiance et aucun d'entre eux n'a fait la moindre erreur. Cela ne rend pas la personne qui a créé les données responsable de l'utilisation malveillante qui en est faite, mais il est raisonnable d'assumer que ça peut arriver, et de décider en connaissance de cause : les programmes sont faillibles et vulnérables : des humains les ont créés.Une dernière chose très importante : quand le van de la police est devant chez vous, c'est trop tard, aucun chiffrement ne va vous aider à ce point si vous n'avez pas pris les mesures nécessaires au préalable. ▮

REMERCIEMENTS

Merci à tous les membres de Syn2cat pour leur support et leur aide, et aux cobayes ayant participé aux salons vie privée !

RÉFÉRENCES

[1] Submarine Cable Map : **http://www.submarinecablemap.com**

[2] La coupure d'Internet en Égypte, une première mondiale par son ampleur : **http://www.liberation.fr/planete/2011/01/28/la-coupure-d-internet-en-egypte-une-premiere-mondiale-par-son-ampleur_710768**

[3] GCHQ Bude : **https://en.wikipedia.org/wiki/GCHQ_Bude**

[4] GCHQ taps fibre-optic cables for secret access to world's communications : **http://www.theguardian.com/uk/2013/jun/21/gchq-cables-secret-world-communications-nsa**

[5] Lightbeam : **https://www.mozilla.org/en-US/lightbeam/**

[6] Browserleaks : **https://www.browserleaks.com/**

[7] **https://krebsonsecurity.com/2015/05/mobile-spy-software-maker-mspy-hacked-customer-data-leaked/**

[8] HTTPS Everywhere : **https://www.eff.org/https-everywhere**

[9] uBlock Origin : **https://github.com/gorhill/uBlock**

[10] Tails : **https://tails.boum.org/**

[11] RIPE: LIRs : **https://labs.ripe.net/statistics/number-of-lirs**

[12] Tor : **https://www.torproject.org/**

BETTERCRYPTO.ORG : GUIDE DE CRYPTOGRAPHIE APPLIQUÉE

David DURVAUX

Besoin de configurer un serveur Apache avec support d'HTTPS ? La cryptographie c'est trop complexe ? Quels sont les choix à faire ? Quels algorithmes ? BetterCrypto.org est un projet open source destiné à vous aider !

Jeudi 6 juin 2013, Edward Snowden commence la publication de documents révélant les pratiques de la NSA. Survient alors une prise de conscience de l'importance de sécuriser les données. Ces révélations confirment que certains algorithmes et protocoles, tels que PGP/GPG ou AES, sont résistants aux attaques.

Le problème suivant se pose alors : comment utiliser correctement ces algorithmes ? Si théoriquement ils sont prouvés mathématiquement, et avérés robustes, leurs mauvaises utilisations peuvent les rendre complètement vulnérables. Et cela sans compter les risques liés à de mauvaises implémentations, ou des bugs oubliés tels que ceux ayant conduit à *HeartBleed*.

Partant de ce constat, et pour répondre aux nombreuses questions d'administrateurs système, le projet BetterCrypto (**[BETTERCRYPTO]**) est né. Le principe est simple : réunir un groupe d'experts avec différentes expériences pour écrire, dans un document de référence, les configurations qui leur semblent optimales. Toutes les décisions sont prises en toute transparence et détaillées dans le document afin que tout un chacun puisse les critiquer et décider en connaissance de cause. C'est ce projet que nous allons vous présenter tout au long de cet article.

La vocation du projet est d'atteindre tous ceux qui ont besoin de protéger leurs données. Pour la plupart des entreprises et des particuliers, une utilisation correcte des protocoles standards est suffisante. A contrario, les organisations aux besoins spécifiques, et disposant d'experts capables de comprendre les subtilités de chaque algorithme trouveront certainement certaines recommandations insuffisantes ou incomplètes.

1. CONSTRUCTION DU DOCUMENT BETTERCRYPTO

Nous nous sommes écartés du schéma classique dans lequel on commence par un long chapitre discutant des tenants et aboutissants de la cryptographie, des algorithmes et de tous les aspects influençant la sécurité.

Partant de l'hypothèse que ce document sera probablement consulté pour la première fois dans l'urgence, nous l'avons construit de telle sorte que la pratique précède la théorie. Après une brève introduction, les configurations de références sont présentées. Ainsi, nos lecteurs peuvent commencer par copier/coller et obtenir rapidement un système fonctionnel et sécurisé.

Le lecteur est toutefois invité à consulter, par la suite, la partie théorique décrivant les choix établis et les raisons qui nous ont poussés à les faire. Il ne tient qu'à eux de décider si la configuration proposée satisfait à leurs besoins ou si elle doit être amendée. Les utilisateurs plus curieux, ou moins pressés, peuvent commencer par la lecture de la partie théorique.

Dans tous les cas, nous espérons qu'ils consulteront les deux sections, et plus encore, qu'ils critiqueront nos décisions.

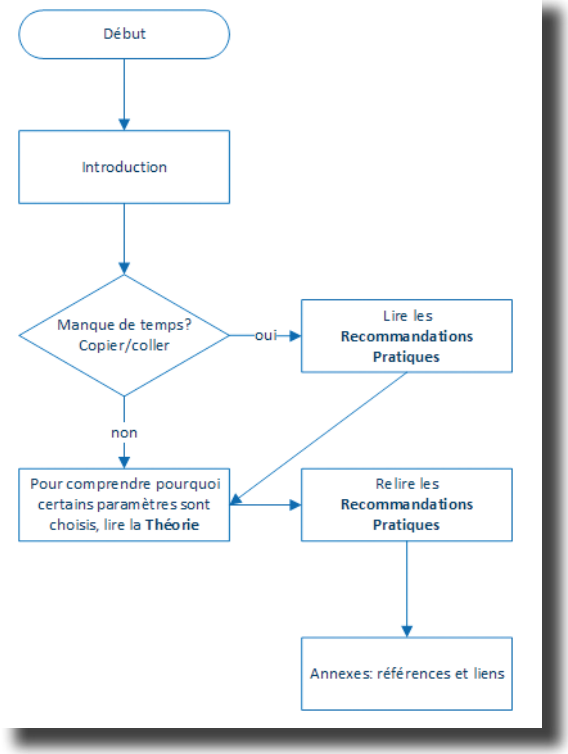


Fig. 1 : Schéma d'utilisation du document. Selon le temps disponible, on commence par copier/coller ou par lire les justifications des choix pris par les auteurs.

2. PARTIE THÉORIQUE

BetterCrypto n'est pas un cours de cryptographie, aussi ne vous attendez pas à y trouver de longues explications sur les algorithmes. Par contre, nos objectifs y sont expliqués et discutés. Nous y présentons en particulier les notions de « *Forward Secrecy* », de compatibilités, de génération de nombres aléatoires, d'échange de clés...

Prenant en compte les différentes contraintes existantes, nous en sommes arrivés à faire deux propositions de choix d'algorithmes. Une très stricte, la suite A, mais avec quelques désavantages, comme l'incompatibilité avec Java 6 et Windows XP, et une suite d'algorithmes plus souple, la suite B, mais dérogeant à certains de nos souhaits comme nous le verrons par la suite.

2.1 Perfect Forward Secrecy

Le *perfect forward secrecy* ([HAC-FS-CHAP12]) est une propriété qui garantit la confidentialité des échanges passés même si les clés du client et/ou serveur sont compromises.

En pratique, cette propriété est atteinte en générant régulièrement une nouvelle clé, par exemple à l'aide de fonctions non-inversibles, afin de chiffrer le trafic. Dès lors, même si l'attaquant arrive à obtenir la clé d'échange courante, les clés précédentes, et donc le contenu chiffré, ne peuvent pas être retrouvés.

Attaché à garantir la confidentialité des communications, c'est un principe important qui n'est malheureusement complètement disponible que dans la suite A.

2.2 Configurations proposées

Dans un monde idéal, seuls les chiffrements permettant de garantir le *perfect forward secrecy*, c'est-à-dire utilisant un échange de clés avec la version éphémère de *Diffie-Helleman* (EDH ou EECDH) combiné avec une authentification forte (telle que RSA) devraient être utilisés. Cette version sans compromis est la suite A qui ne contient que 4 algorithmes :

- EDH, aRSA et AES128 ;
- EDH, aRSA et AES256 ;
- EECDH, aRSA et AES128 ;
- EECDH, aRSA et AES256.

En notation *OpenSSL*, nous avons la chaîne de caractères suivante (notez l'exclusion explicite de SSL en version 3) :

Fichier

```
EDH+aRSA+AES128:
EDH+aRSA+AES256:EECDH+aRSA+AES256:EECDH+aRSA+AES256:!SSLv3
```

En pratique, cette suite va poser problème si vous utilisez des logiciels ne supportant pas TLS 1.2. Il en va de même si vos équipements disposent de

À savoir

La différence entre les versions EDH ou EECDH de *Diffie-Helleman* réside dans l'utilisation ou non des courbes elliptiques. Alors que la version EDH utilise de grands nombres premiers, la version EECDH va utiliser des courbes elliptiques, ce qui permet de maintenir un même niveau de sécurité pour un coût de calcul plus faible.

ressources limitées, car l'utilisation de TLS 1.2 peut avoir des conséquences non négligeables sur la consommation de ressources matérielles.

Afin de résoudre ce problème, nous avons étendu la suite A avec les versions 1.0 et 1.1 de TLS, et ajouté SHA1 à la liste des protocoles utilisés pour l'authentification des paquets.

Nous obtenons ainsi la suite B, décrite avec la chaîne de caractères suivante pour OpenSSL :

Fichier

```
EDH+CAMELLIA:EDH+aRSA:EECDH+aRSA+AESGCM:EECDH+aRSA+SHA256:EECDH:+CAMELLIA1
28:+AES128:+SSLv3:!aNULL:!eNULL:!LOW:!3DES:!MD5:!EXP:!PSK:!DSS:!RC4:!SEED:
!IDEA:!ECDSA:kEDH:CAMELLIA128-SHA:AES128-SHA
```

Notez que les protocoles considérés comme faibles ou vulnérables sont explicitement exclus. Nous retrouvons dans cette liste d'exclusion 3DES, MD5, RC4... Cet ajout d'algorithmes supplémentaires permet de supporter Windows XP ainsi que d'autres logiciels ou systèmes d'exploitation vieillissants.

2.3 Longueurs des clés

Quelle est la longueur de clés à recommander ? Cette question nous a occupés pendant quelque temps et de nombreuses discussions ont eu lieu autour d'AES. Si intuitivement AES256 semble mieux que AES128 (même si un article de Bruce Schneier pourrait contredire cette affirmation **[SCHNEIER-AES]**), la réponse finale concernant AES est venue de Vincent Rijmen – un des auteurs de l'AES – dans l'affirmation suivante :

« *On the choice between AES256 and AES128: I would never consider using AES256, just like I don't wear a helmet when I sit inside my car. It's too much bother for the epsilon improvement in security.* »

Contenu d'un échange privé de courriel avec Vincent Rijmen en décembre 2013.

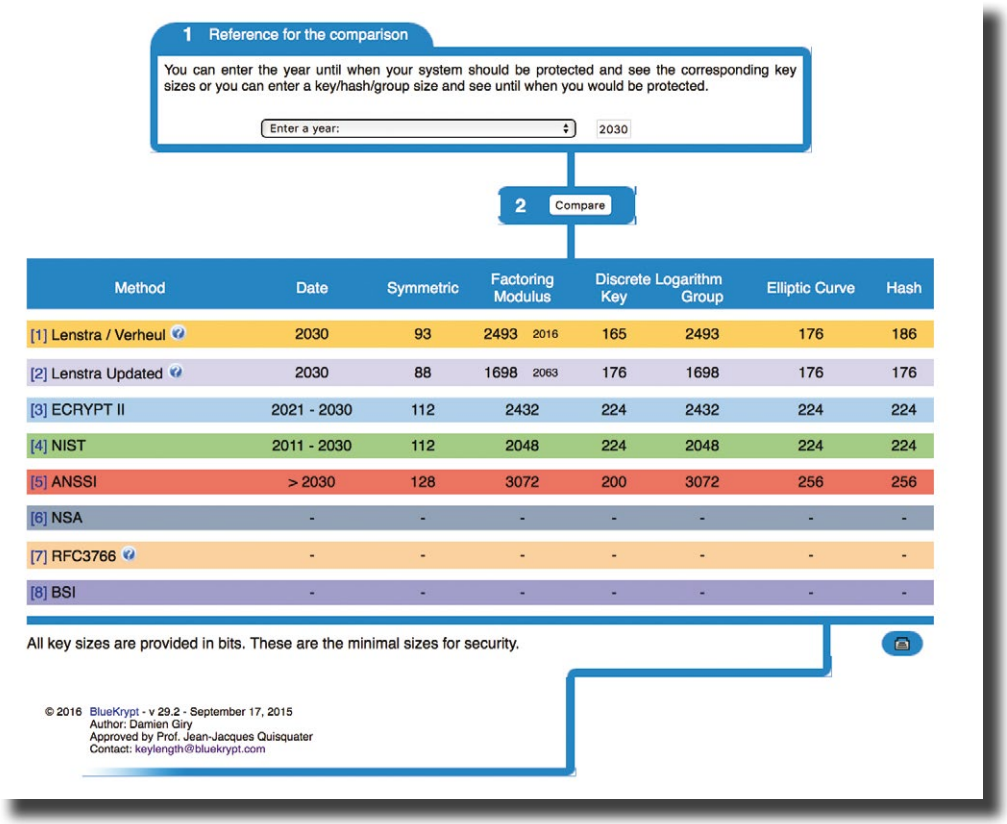
Comme aucun de nous ne semblait disposé à conduire nos voitures avec un casque, nous avons suivi sa recommandation et inclus AES128.

De manière plus rigoureuse, nous avons pris en compte le système de comparaison proposé par BlueKrypt sur son site **[KEYLENGTH]** (Figure 2, page suivante).

Cette référence compare différentes normes (ANSSI, NIST...) pour faciliter le choix de la longueur de clé idéale selon des critères objectifs comme, par exemple, la durée pendant laquelle les informations devront rester protégées.

Pour notre projet, nous avons pris les recommandations suivantes pour la longueur des clés :

- Chiffrement asymétrique (comme RSA) : 3248 bits (recommandation Ecrypt II) ;
- Chiffrement symétrique (tel que AES) : 128 bits ;
- Chiffrement utilisant les courbes elliptiques (ECC) : 256 bits ;
- Algorithme de hash : minimum SHA2+ (SHA256 par exemple).



Method	Date	Symmetric	Factoring Modulus	Discrete Logarithm Key	Discrete Logarithm Group	Elliptic Curve	Hash
[1] Lenstra / Verheul	2030	93	2493 2016	165	2493	176	186
[2] Lenstra Updated	2030	88	1698 2063	176	1698	176	176
[3] ECRYPT II	2021 - 2030	112	2432	224	2432	224	224
[4] NIST	2011 - 2030	112	2048	224	2048	224	224
[5] ANSSI	> 2030	128	3072	200	3072	256	256
[6] NSA	-	-	-	-	-	-	-
[7] RFC3766	-	-	-	-	-	-	-
[8] BSI	-	-	-	-	-	-	-

Fig. 2 : Comparaison des différentes normes selon un critère choisi : ici, les longueurs de clés nécessaires pour garantir la protection jusqu'en 2030.

2.4 Compatibilité

Le choix entre les suites A et B se fera essentiellement sur base des logiciels qui doivent être supportés. S'ils sont tous récents et sous contrôle, la suite A offrira une sécurité accrue. Si, au contraire, des configurations plus anciennes ou plus exotiques doivent être supportées alors la suite B offrira un compromis acceptable.

De manière générale, pour des systèmes à usage interne et/ou sensible (même si accessible depuis l'Internet), la suite A devra être préférée. A contrario pour un système grand public, la suite B réduira le nombre de clients frustrés par les connexions refusées et les messages d'erreur de leurs navigateurs.

De manière non exhaustive, le tableau suivant compare les configurations supportées par les suites A et B. Avant toute mise en production, certains outils permettent de définir de manière plus précise les configurations compatibles. Nous y reviendrons plus tard (Tableau 1, ci-contre).

En résumé, les machines exécutant au minimum Windows 7, Mac OS X 10.9 (Mavericks), Android 4.4 (KitKat), Windows Phone 8.1 ou iOS 6 devraient supporter les suites A et B si les logiciels ont bien été mis à jour aux dernières versions disponibles. Les logiciels plus vieux seront incompatibles avec TLS 1.2 et donc avec la suite A.

2.5 Génération de nombres aléatoires

Une des hypothèses de base utilisées en cryptographie est la capacité de générer des nombres aléatoires. Pour ce faire, de multiples techniques sont combinées afin d'obtenir ce qui se rapprochera le plus d'un générateur parfait de nombres aléatoires.

Configuration	Suite A	Suite B
Android 2.3 – Android 4.3	Non	Oui
Android 4.4.2 et supérieur	Oui	Oui
Google Chrome 47	Oui	Oui
Firefox 31.0.3 ESR	Oui	Oui
Firefox 42	Oui	Oui
Internet Explorer 6	Non	Non
Internet Explorer 7	Non	Oui
Internet Explorer 8 à 10	Non	Oui
Internet Explorer 11 et supérieur	Oui	Oui
Windows Phone 8.0	Non	Oui
Windows Phone 8.1 et supérieur	Oui	Oui
Edge	Oui	Oui
Java 6 à 7	Non	Oui (DH limité à 1024 bits)
Java 8	Oui (DH limité à 1024 bits)	Oui (DH limité à 1024 bits)
OpenSSL avant version 1.0	Non	Oui
OpenSSL 1.0 et supérieur	Oui	Oui
Safari 5.1.9 et 6.04	Non	Oui
Safari 7 et supérieur	Oui	Oui
IOS 6.0.1	Oui	Oui

Toutefois, la popularité grandissante des appareils mobiles pose problème. Les clés sont souvent générées au démarrage de l'appareil alors que la réserve d'entropie est faible. Il est dès lors possible de prédire la suite de valeurs aléatoires générées, surtout si tous partagent les mêmes paramètres d'initialisation.

Une bonne pratique est donc de renouveler les clés régulièrement, et surtout, d'éviter de les générer pendant, ou immédiatement après la phase de démarrage. Le but étant d'assurer que votre système dispose de suffisamment d'entropie pour réduire au minimum les risques de prédiction de la suite des nombres aléatoires.

À retenir

À titre d'exemple, un article de la base de connaissances de Juniper Networks d'avril 2010 est intéressant ([JUNIPER_KB]). Il explique qu'en raison de la manière dont l'entropie est obtenue, il existe un risque que plusieurs routeurs partagent la même clé privée pour SSH.

3. PARTIE PRATIQUE

Lister de manière exhaustive tous les logiciels supportés par ce projet serait inutile, car cela serait trop long ou incomplet selon les opinions. Nous avons néanmoins tenté de couvrir un maximum de cas et nous continuerons à le faire dans la mesure du possible.

Citons toutefois les grands classiques tels que :

- les serveurs web : Apache, ligthttpd, nginx et Microsoft IIS ;
- le shell sécurisé : OpenSSH ;

- les serveurs de courriels : Dovecot, Cyrus, Postfix et exim ;
- les VPN : IPSec, OpenVPN, CheckPoint et CISCO ASA ;
- l'encryption des courriels avec PGP/GPG ;
- les serveurs de messagerie instantanée : ejabberd et Charybdis ;
- les gestionnaires de bases de données : Oracle, MySQL, DB2 et PosgreSQL ;
- les proxy : BlueCoat, HAProxy, Pound et stunnel ;
- l'authentification avec Kerberos.

Pour chacun, nous ne pouvons que vous inviter à consulter le projet BetterCrypto. Nous ne nous attarderons ici que sur Apache et Postfix pour la suite de cette partie pratique.

Comme vous le constaterez, les changements à apporter aux configurations sont détaillés. Lorsque la configuration ne doit pas être adaptée, il en est fait mention dans le projet, mais aucune configuration ne sera proposée. À terme, notre rêve serait de pouvoir dire que par défaut, tout est correct et bien configuré. En pratique, on est relativement proche de cette situation idéale pour un certain nombre d'outils commerciaux et open source.

3.1 Apache

Apache est un serveur web open source que l'on rencontre très fréquemment sur la toile, tellement populaire que son nom fait partie des acronymes LAMP (Linux – Apache – MySQL – PHP) ou WAMP (Windows – Apache – MySQL – PHP).

Il sera donc notre cobaye pour cette première démonstration du contenu du projet BetterCrypto.

La première partie de la configuration proposée consiste à fournir la configuration de base pour SSL :

Fichier

```
SSLCertificateFile /etc/ssl/certs/ssl-cert-snakeoil.pem
SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key
#SSLCertificateChainFile /etc/apache2/ssl.crt/server-ca.crt
#SSLCACertificateFile /etc/apache2/ssl.crt/ca-bundle.crt
SSLProtocol All -SSLv2 -SSLv3
SSLHonorCipherOrder On  SSLCompression off


# Add six earth month HSTS header for all users...
Header always set
Strict-Transport-Security "max-age=15768000"


# If you want to protect all subdomains, use the following header
# ALL subdomains HAVE TO support HTTPS if you use this!
# Strict-Transport-Security: "max-age=15768000 ; includeSubDomains"
# HTTP Public Key Pinning (HPKP) for 90 days (60*60*24*90=7776000)
# At least use one Backup-Key and/or add whole CA, think of Cert-Updates!
Header always set
Public-Key-Pins "pin-sha256=\"YOUR_HASH=\"; pin-sha256=\"\
```

```
 \YOUR_BACKUP_HASH=\"; max-age=7776000; report-uri=\"https://YOUR.REPORT.
URL\""

SSLCipherSuite 'EDH+CAMELLIA:EDH+aRSA:EECDH+aRSA+AESGCM:EECDH+aRSA+SHA256:
EECDH\
 \:+CAMELLIA128:+AES128:+SSLv3:!aNULL:!eNULL:!LOW:!3DES:!MD5:!EXP:!PSK:!DSS\
 \:!RC4:!SEED:!IDEA:!ECDSA:kEDH:CAMELLIA128-SHA:AES128-SHA'
```

Plusieurs choses sont à noter sur cette configuration :

- Elle fournit les éléments à ajouter à une configuration standard pour obtenir une configuration complète supportant HTTPS. Le but est d'aider à la transition vers HTTPS, pas d'écrire un guide sur Apache. Ces derniers existent déjà en suffisance et en qualité.
- Le paramètre SSLCipherSuite sert à définir la suite de chiffrement. On reconnaît tout de suite la présence de la suite B dans cet exemple.
- HSTS (*HTTP Strict Transport Security*) **[HSTS-Wikipedia]**, est activé. Cette directive, envoyée par le serveur, demande au navigateur de communiquer avec le serveur uniquement au travers d'une communication sécurisée.

Pour ceux qui veulent laisser le port TCP/80 ouvert, tout en forçant le passage vers HTTPS, la configuration suivante peut être donnée à Apache :

Fichier

```
<VirtualHost *:80>
 Redirect permanent / https://SERVER_NAME/
</VirtualHost>
```

Des configurations équivalentes sont proposées pour les autres serveurs web populaires.

3.2 Postfix

Le cas du serveur de courriels est très intéressant : non seulement il doit être compatible avec les logiciels de messagerie qui viennent recevoir ou envoyer des courriels, mais il doit être capable de servir de relais vers d'autres serveurs. Si on peut parfois accepter qu'un client soit incompatible, afin d'inciter l'utilisateur à mettre à jour son logiciel, cela devient inacceptable lorsque la communication doit être absolument établie entre deux serveurs.

Nous distinguons trois modes de fonctionnement des serveurs de messagerie :

- mode soumission (MSA) : l'utilisateur y envoie ses courriels le plus souvent en utilisant le protocole SMTP ;
- mode transfert (MTA) ou échange (MX) : les courriels sont échangés entre les serveurs source et destination en utilisant également le protocole SMTP ;
- mode réception (MDA) : l'utilisateur vient récupérer ses courriels le plus souvent via les protocoles IMAP ou POP.

Il est également à noter que pour les serveurs de courriels, TLS peut être utilisé à deux niveaux : soit en « emballage » du protocole (POPS, IMAPS et SMTPS), soit dans les protocoles en texte clair, en émettant la commande STARTTLS. Ce second mode, appelé TLS opportuniste, est très intéressant dans les communications entre serveurs, car il permet d'utiliser une communication chiffrée si les deux parties le supportent. Si l'une des deux n'implémente pas la commande, la communication continue sans chiffrement. Ce mode est en cours de déploiement depuis environ deux ans et a provoqué une augmentation massive du chiffrement des échanges SMTP initiant une connexion SSL (**[FACEBOOK] [GOOGLE-EMAIL]**).

3.2.1 Mode SMTP

Pour les communications via SMTP, nous recommandons de ne pas affecter les suites de chiffrement par défaut de Postfix. Au vu de l'hétérogénéité des configurations sur Internet, il est ici souhaitable d'être capable de supporter tous les cas de figure afin de pouvoir délivrer les courriels de l'utilisateur.

Cela amène d'ailleurs à deux points d'attention :

- D'une part à l'importance d'éduquer les utilisateurs aux risques de transmettre des informations par courriel ;
- D'autre part à l'existence de techniques de chiffrement du contenu des courriels pour en protéger le contenu et/ou en garantir l'intégrité (nous parlons ici des chiffrements S/MIME ou PGP/GPG).

La configuration que nous retrouvons est une configuration standard, si ce n'est qu'elle active le mode opportuniste pour TLS.

Fichier

```
# TLS parameters
smtpd_tls_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
smtpd_tls_key_file=/etc/ssl/private/ssl-cert-snakeoil.key

# use 0 for Postfix >= 2.9, and 1 for earlier versions
smtpd_tls_loglevel = 0
# enable opportunistic TLS support in the SMTP server and client
smtpd_tls_security_level = may
smtp_tls_security_level = may
smtp_tls_loglevel = 1

# if you have authentication enabled, only offer it after STARTTLS
smtpd_tls_auth_only = yes
tls_ssl_options = NO_COMPRESSION
```

3.2.2 Mode MSA

En mode MSA, le client de l'utilisateur communique directement avec le serveur. On peut dès lors se permettre de forcer les algorithmes qui seront employés. Dans le fichier de configuration **`main.cf`**, on définit que le chiffrement est obligatoire, que les protocoles faibles sont bannis (SSLv2 et SSLv3) et on précise la suite de chiffrement qui sera proposée à la négociation.

Fichier

```
smtpd_tls_mandatory_protocols = !SSLv2, !SSLv3
smtpd_tls_mandatory_ciphers=high tls_high_cipherlist=EDH+CAMELLIA:EDH+aRSA:
EECDH+aRSA+AESGCM:EECDH+aRSA+SHA256:\
 \EECDH:+CAMELLIA128:+AES128:+SSLv3:!aNULL:!eNULL:!LOW:!3DES:!MD5:!EXP:!
PSK:!DSS\
 \:!RC4:!SEED:!IDEA:!ECDSA:kEDH:CAMELLIA128-SHA:AES128-SHA
```

À nouveau, la suite de chiffrement B y a été ici choisie.

Optionnellement, pour les utilisateurs des courbes elliptiques, on peut adapter le paramètre spécifique dans **`main.cf`** (il est défini par défaut à « strong ») :

Fichier

```
smtpd_tls_eecdh_grade=ultra
```

Pour compléter la configuration, il est encore nécessaire de définir les paramètres propres au mode MSA dans le fichier **`master.cf`** :

Fichier

```
submission inet n - - - - smtpd  -o smtpd_tls_security_level=encrypt
-o tls_preempt_cipherlist=yes
```

4. LES ATTAQUES CONNUES ET NOS RÉSULTATS

Depuis les débuts du projet, plusieurs attaques ont été découvertes. Citons, par exemple, HeartBleed, Poodle, Freak, Logjam et, tout récemment, Drown ([TLS-SEC]).

Nous pouvons tirer deux conclusions de ces attaques (méconnues lors de l'écriture de la première version du projet) :

- il est important de valider les implémentations autant que les principes théoriques ;
- si certains paramètres permettaient d'éviter l'attaque, ils faisaient partie des recommandations.

4.1 POODLE

Cette attaque fonctionne en faisant un « *man-in-the-middle* » pour rétrograder le chiffrement vers SSLv3 ([POODLE]). L'attaquant procède alors à de la duplication de blocs. En moyenne, pour 256 requêtes, il pourra déchiffrer 1 byte.

Pratiquement, cette attaque de 2014 n'est possible que si SSLv3 est autorisé. Notre recommandation de désactiver SSLv3 a rendu l'attaque sur le *padding* impossible.

Il est à noter que certaines implémentations de TLS étaient quand même vulnérables à POODLE. Le fait que les mises à jour aient corrigé cette faille montre combien il est important de mettre les systèmes d'exploitation à jour.

4.2 TLS Logjam

Cette attaque d'octobre 2015 exploite une vulnérabilité dans le protocole TLS ([WEAKDH]). L'attaque se fait de deux manières différentes au niveau de Diffie-Helleman :

1. Soit en exploitant le support des vieilles suites de chiffrement pour forcer l'utilisation d'algorithmes obsolètes qui utilisent des clés de 512 bits ;
2. Soit en exploitant le fait que, par défaut, des millions de serveurs HTTPS, SSH ou VPN utilisent les mêmes nombres premiers pour l'échange de clés avec Diffie-Helleman.

En configurant la version éphémère de Diffie-Helleman ou des nombres premiers de plus de 1024 bits, l'attaque est évitée. Cela correspond aux recommandations du projet, et cela bien avant l'attaque.

5. COMMENT TESTER ?

Après avoir défini de manière théorique les paramètres que nous voulions, de multiples tests ont été effectués pour valider la praticabilité de ceux-ci :

- ⇨ compatibilité avec les différentes versions de logiciels (les versions testées sont référencées dans le document) ;

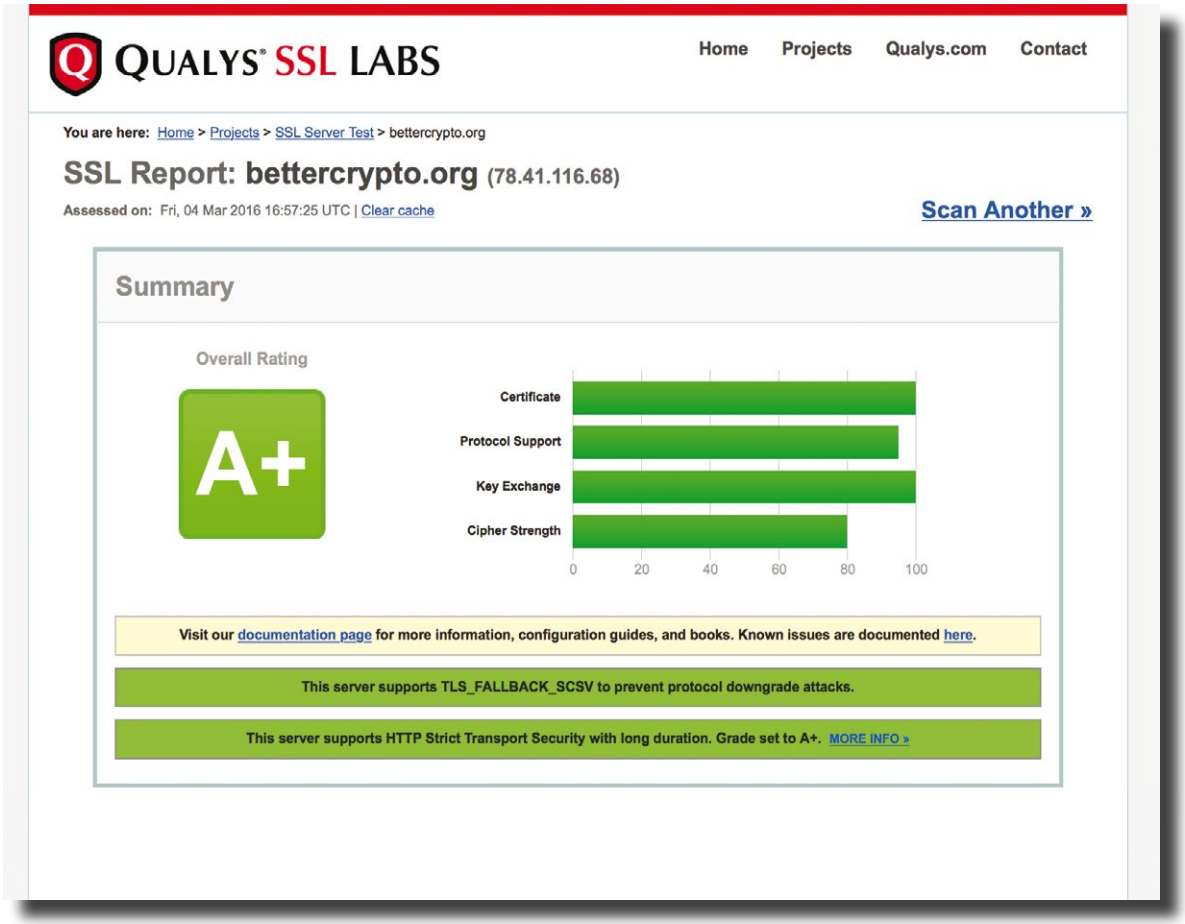


Fig. 3 : SSLLabs sur BetterCrypto.org.

- compatibilité de ces choix avec les clients courants ;
- robustesse des choix effectués.

Nous avons testé nos configurations avec de nombreux outils qui sont décrits dans BetterCrypto. Toutefois, l'un d'entre eux mérite d'être mentionné ici. Qualys propose un outil, en ligne, pour tester la robustesse de vos sites Internet. *SSLLabs* (**[SSLLABS]**) fait un test exhaustif des algorithmes supportés et produit un rapport incluant les clients qui seront supportés et les éventuelles faiblesses de la configuration choisie. C'est un outil qui devrait être utilisé systématiquement et régulièrement par tous ceux qui déploient des sites Internet.

L'annexe A du projet BetterCrypto (**[BETTERCRYPTO]**) propose une liste d'outils pour tester les différents éléments qui feront le succès de la sécurisation des échanges : tests des serveurs et des navigateurs, longueurs des clés et génération de nombres aléatoires.

CONCLUSION

Si vous consultez le document, vous constaterez que nous avons laissé un marquage « DRAFT ». Même s'il est relativement complet, il continue à évoluer en permanence.

De nombreux projets sont encore sur la table. Nous aimerions écrire un générateur de configurations supportant davantage de logiciels et de versions de ceux-ci. Nous voudrions également ajouter plus de logiciels commerciaux tels que les produits Microsoft par exemple.

Un nouveau format est également envisagé. Nous produisons actuellement un document PDF qui semble ne pas être l'optimum pour copier/coller. Le générateur de configuration serait8 certes une solution, mais pas complètement satisfaisante. Une réflexion sera probablement entamée sur le meilleur moyen de fournir le contenu pour que son accès soit le plus simple et pratique possible.

Même les choix théoriques sont remis en question régulièrement. Au moment d'écrire ces lignes, des discussions ont lieu sur notre liste de diffusion (**[BT-ML]**) concernant les suites A et B. La suite A est remise en question, car peu utilisée . Camellia pourrait également être voué à la suppression de la suite B car, autant AES est étudié et attaqué par de nombreux académiques, autant Camellia est pour le moment peu évalué.

Le projet est publié avec une licence *Creative Commons Attribution-ShareAlike* (**[CC BY-SA-3.0]**). Vous êtes donc les bienvenus pour apporter votre contribution : que cela soit en utilisant notre projet, en relisant, en proposant de nouvelles sections... Si cela vous tente, n'hésitez pas à rejoindre notre liste de diffusion et à nous contacter.

Finalement, 30 mois après la création du projet, son utilité semble avoir été démontrée à plusieurs reprises. Bien qu'imparfait, il a pour vocation de simplifier l'utilisation correcte de la cryptographie. Si quelques personnes appliquent avec succès nos recommandations, alors ce projet sera un grand succès. ■

REMERCIEMENTS

Cet article n'a pu être écrit que grâce à la collaboration de toutes les personnes ayant participé de près ou de loin à ce projet. Et la liste est longue ! Auteurs, lecteurs, critiques, enseignants... nombreux sont ceux qui ont donné leurs avis, leurs commentaires et leurs corrections.

Ce projet est, en particulier, le fruit de la passion d'Aaron Kaplan et Pepi Zawodksy. Ils méritent une mention et un merci tout particulier,

Finalement, impossible de ne pas remercier *MISC* qui nous offre une opportunité exceptionnelle de présenter notre travail.

RÉFÉRENCES

⇨ [HAC-FS-CHAP12] A. Menezes, P. van Oorschot, and S. Vanstone, Chapitre 12 de « Handbook of Applied Cryptography », CRC Press, 1996 :
http://cacr.uwaterloo.ca/hac/about/chap12.pdf

⇨ [SCHNEIER-AES] Another New AES Attack :
https://www.schneier.com/blog/archives/2009/07/another_new_aes.html

⇨ [BETTERCRYPTO] Site officiel du projet :
https://www.bettercrypto.org

⇨ [BC-GIT] Miroir du projet sur GitHub :
https://github.com/BetterCrypto/

⇨ [KEYLENGH] Cryptographic Key Length Recommendation :
https://www.keylength.com

⇨ [JUNIPER_KB] Multiple routers can generate duplicate SSH private keys due to missing entropy :
https://kb.juniper.net/InfoCenter/index?page=content&id=JSA10434&actp=search

⇨ [HSTS-Wikipedia] HTTP Strict Transport Security :
https://fr.wikipedia.org/wiki/HTTP_Strict_Transport_Security

⇨ [FACEBOOK] The Current State of SMTP STARTTLS Deployment :
https://www.facebook.com/notes/protect-the-graph/the-current-state-of-smtp-starttls-deployment/1453015901605223/

⇨ [GOOGLE-EMAIL] Email encryption in transit :
https://www.google.com/transparencyreport/saferemail/

⇨ [TLS-SEC] An introduction to and survey of TLS Security :
http://www.slideshare.net/a_z_e_t/introduction-to-and-survey-of-tls-security-bsideshh-2014

⇨ [POODLE] This POODLE Bites: Exploiting The SSL 3.0 Fallback :
https://www.openssl.org/~bodo/ssl-poodle.pdf

⇨ [WEAKDH] Weak Diffie-Hellman and the Logjam Attack :
https://weakdh.org

⇨ [SSLLABS] Qualys SSL Labs : **https://www.ssllabs.com/**

⇨ [BT-ML] Ach -- Applied Crypto Hardening list :
http://lists.cert.at/cgi-bin/mailman/listinfo/ach

Quarkslab

SECURING EVERY BIT OF YOUR DATA

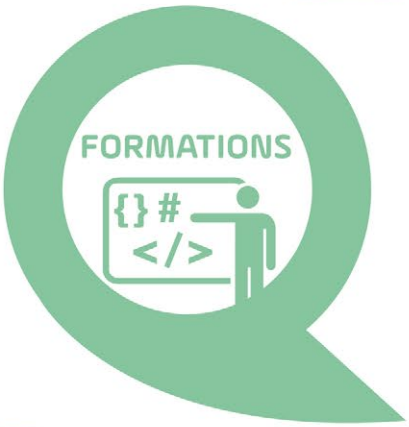
Les attaquants ciblent les données, et non les infrastructures qui sont régulièrement surveillées, testées et mises à jour. Quarkslab se concentre sur la sécurisation des données, au travers de 3 outils issus de notre R&D : Cappsule (hyperviseur), IRMA (analyseur de fichiers) et Epona (obfuscateur). Ces produits, qui complètent nos services et formations, visent à aider les organisations à prendre leurs décisions au bon moment grâce à des informations pertinentes.

Cappsule virtualise instantanément et sans intervention toutes vos applications à la volée pour cloisonner les données.

IRMA analyse des fichiers pour déterminer leur dangerosité, et fournit une vue détaillée des incidents détectés.

Epona obfusque du code pour contrarier le reverse engineering et l'accès aux données des applications.

- **Tests de sécurité :** analyse d'applications, de DRM, de vulnérabilités, de patch, fuzzing
- **Développement & analyse :** R&D à la demande, reverse engineering, design et implémentation
- **Cryptographie :** conception de protocoles, optimisation, évaluation



- Reverse engineering
- Recherche de vulnérabilités
- Développement d'exploits
- Test de pénétration d'applications Android / iOS
- Windows internals

Quarkslab
SECURING EVERY BIT OF YOUR DATA
71 Avenue des Ternes - 75017 Paris - FRANCE
Phone: +33 (0)1 56 60 21 02 - Email: contact@quarkslab.com
@quarkslab - **www.quarkslab.com**

2

2 LES TIERS DE CONFIANCE EN QUESTION

À découvrir dans cette partie...

2.1 Souriez ! Les autorités de certification sont filmées !

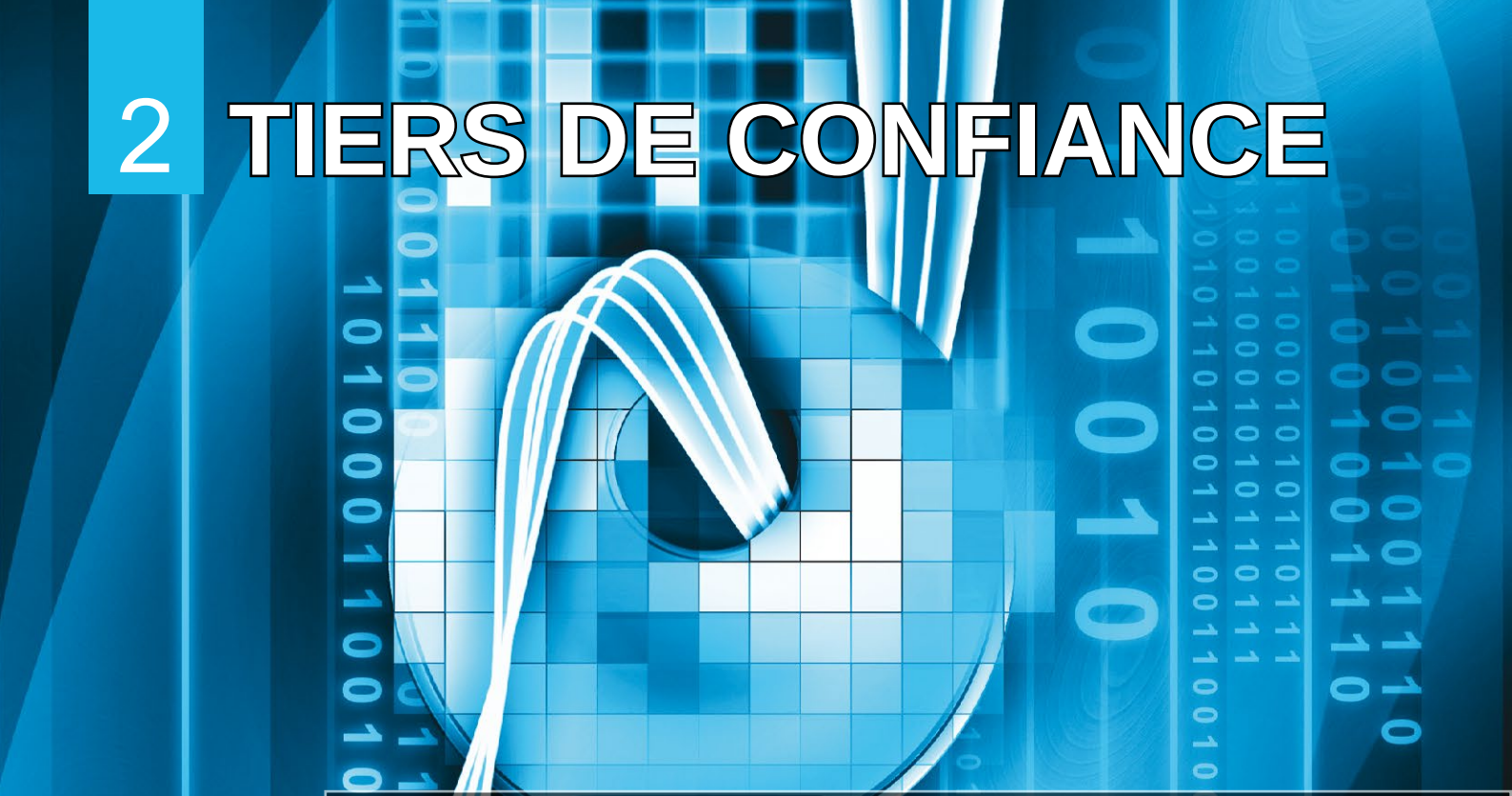
Qui n'a pas déjà entendu une histoire d'une autorité de certification ayant émis un certificat frauduleusement ? Le protocole certificate transparency vise à améliorer le système de réputation des CAs en permettant de surveiller et auditer leur activité. Découvrez son fonctionnement. p. 50

2.2 Letsencrypt : HTTPS et TLS enfin industrialisés

Qui souhaite déployer un site derrière HTTPS doit passer obligatoirement par la case : acheter et créer un certificat X509. Let's encrypt permet désormais d'automatiser au maximum ce processus, et cela sans avoir à verser un seul denier ! p. 60

2.3 Détecter l'interception des flux web chiffrés

Découvrez comment fonctionnent les interceptions TLS classiques, en entreprise comme chez vous, ainsi que les manières de les détecter. p. 70



SOURIEZ ! LES AUTORITÉS DE CERTIFICATION SONT FILMÉES !

Florian MAURY

Il y a peu, chacun vivait dans son petit village et n'avait qu'une poignée de pairs à qui devoir faire confiance. Internet a révolutionné ce modèle : nous interagissons tous quotidiennement avec un nombre de tiers, dépassant notre capacité de jugement. Comment alors faire passer à l'échelle d'Internet notre confiance en autrui ? Cet article détaille *Certificate Transparency*, un mécanisme cryptographique enregistrant les actions critiques effectuées par les autorités de certification. Équipée de cet outil, la société connectée peut alors détecter les erreurs et les trahisons, lorsqu'elles surviennent, améliorer son système de réputation et accroître ses capacités à punir les contrevenants.

1. LA CONFIANCE ET LES PRESSIONS SOCIÉTALES

1.1 La confiance sous l'angle théorique

La confiance peut sembler fondamentalement irrationnelle. Tout individu est animé par des intérêts propres, qu'il place naturellement au-dessus de ceux des autres. Comment donc faire confiance à autrui, sachant qu'à la moindre occasion, celui-ci pourrait nous trahir ? La réponse à cette question n'est pas simple. Les psychologues, les sociologues et les théoriciens du jeu **[poule]** s'interrogent ainsi à ce sujet depuis des siècles.

La civilisation nous a cependant montré qu'une collaboration des parties amenait généralement de meilleurs résultats. Rien de nouveau pour l'Homme moderne, qui sait depuis la préhistoire que la chasse en groupe permet simultanément de chasser des plus gros gibiers et d'améliorer le sens tactique de la tribu, en vue de conflits armés ultérieurs. La civilisation implémente donc, sciemment ou non, pour son bien commun, de multiples pressions sociétales visant à encourager la collaboration, et à punir les défections. Bruce Schneier développe ainsi dans son livre *Liars and Outliers* **[Schneier]** un ensemble de classifications qui se résument grossièrement comme suit :

- la morale régule essentiellement les actions dans de petits groupes d'individus. Plus l'anonymat se renforce et plus les règles morales sont violées ;
- la réputation permet de garantir la collaboration dans un groupe de taille modérée. Ce système permet de suivre la confiance globalement tenue par la société envers les individus. La réputation permet ainsi d'augmenter la taille des cercles de confiance en créant des stéréotypes et des classes de confiance dans lesquels il est possible de ranger les membres de la société ;
- les institutions régulent les groupes de tailles supérieures. Il est admis par certains sociologues que le seuil se situe autour du nombre de Dunbar **[Dunbar]** – environ 150 membres. L'institution vise à définir des règles sociétales pour lesquelles les sanctions, en cas de défection, sont codifiées ;
- les mesures de sécurité, qui permettent de restreindre le champ d'action des fauteurs et des traîtres, d'augmenter le risque encouru en trahissant, de diminuer le gain potentiel, ou de rendre comptable plus facilement les traîtres.

Les systèmes basés sur la réputation peuvent aussi passer à l'échelle grâce à la notion d'introducteurs de confiance : un système où la créance est transitive. Parce que je fais confiance à Alice, et qu'elle fait confiance à Bob, alors peut-être fais-je confiance à Bob, indirectement. Cette pratique, plus risquée pour un individu, permet cependant de dépasser le seuil du nombre de Dunbar. Reste que le nombre d'introducteurs de confiance doit lui-même rester limité, sous peine de devoir empiler les couches d'indirection de la confiance.

1.2 La confiance sous l'angle d'Internet

Les internautes font confiance aux navigateurs web, par nécessité : leurs développeurs détiennent un monopole de fait de la consommation d'information sur le Web.

Les organismes produisant ces navigateurs ont confiance en des autorités de certification (AC), qui servent à leur tour d'introducteurs de confiance pour l'ensemble des sites web. Chaque

navigateur dispose d'un « magasin de certificats d'autorités racines ». Il contient la liste de tiers de confiance d'après ce navigateur. Cette liste varie d'un navigateur à l'autre. Tous les navigateurs requièrent le suivi d'un tronc commun d'exigences, dictées par un consortium : le *CA/Browser Forum* **[CABFORUM]**. Ensuite, chacun rajoute ses bonnes pratiques et des audits réguliers. Seules les AC se pliant à ces demandes sont ajoutées. Certaines sont aussi acceptées, mais annotées de conditions d'utilisation particulières, comme, par exemple, la limitation à certains noms de domaine de premier niveau **[CertLimit]**.

Malgré cela, la presse rapporte régulièrement des attaques réussies ou l'observation de pratiques douteuses chez certaines AC. S'il est normal de regretter la présence de fauteurs, il faut néanmoins savoir raison garder : la présence de traîtres, involontaires ou agissant de façon délibérée, est endémique à tout groupe d'individus et croît mécaniquement avec la taille du groupe. Tant que ces derniers ne sont pas trop nombreux ou trop efficaces, le système reste viable. « Un bon parasite ne tue pas immédiatement son hôte », comme le souligne Schneier.

Doit-on, dès lors, garder la taille de ce groupe modérée, afin de diminuer la probabilité de défection ? Il est notable que les mécanismes de sécurité actuellement implémentés peuvent s'avérer dérisoires si l'on considère que la taille de certains magasins est un multiple du nombre de Dunbar **[EFFObs]** ! Si leur nombre doit pouvoir être réduit, il convient néanmoins de reconnaître la nécessité d'entretenir la concurrence commerciale et une diversité internationale. L'internationalité des AC peut en effet améliorer la qualité des contrôles des documents juridiques locaux prouvant l'identité d'un demandeur de certificat électronique.

Le nombre d'AC pouvant donc être difficile à réduire, il convient d'étudier d'autres pistes. Celle des pressions sociétales institutionnelles semblerait adaptée compte tenu du nombre de partis impliqués. Cette approche semble cependant prématurée, eu égard à la maturité actuelle des systèmes juridiques internationaux sur le droit informatique.

L'amélioration du système de réputation par l'implantation de nouveaux mécanismes de sécurité est une voie envisageable. Une façon de procéder est ainsi d'améliorer la capacité d'observation des défections. En effet, comment affecter la réputation d'un organisme si ses trahisons passent inaperçues ? « Pas vu, pas pris, pas pendu ! » Or, jusqu'à récemment, il était très difficile de détecter une émission de certificats frauduleux. Il fallait prendre l'AC la main dans le sac en constatant l'usage d'un certificat frauduleux lors d'une attaque. Cela nécessitait donc une victime très vigilante ; que l'attaquant s'en prenne à Google, pour lequel Chrome assure une surveillance particulière **[ChromeSurv]** ; ou l'usage d'outils, peu répandus, comme Perspectives **[Perspectives]**. C'est ainsi qu'ont été mises en défaut plusieurs AC, ces dernières années, dont Symantec **[Symantec]**, Turktrust **[ChromeSurv]**, CNNIC **[CNNIC]**, India NIC **[IndiaNIC]**, ou Diginotar **[Diginotar]**.

Certificate Transparency (CT) a été créé en vue d'améliorer la réponse à la problématique de détection des certificats frauduleux *a priori* et d'imputation *a posteriori*. Mais qu'est-ce que CT exactement ?

2. UN JOURNAL GRAVÉ DANS LE MARBRE

Afin d'accompagner le lecteur en douceur, le choix a été fait de séparer ce chapitre en deux sous-sections : d'une part, une description intuitive des interactions entre acteurs et la nature de leurs liens de confiance, et d'autre part, une introduction technique expliquant la cryptographie mise en œuvre.

2.1 Certificate Tranparency, d'un point de vue fonctionnel

Imaginez un village. Le maire a accrédité des fabriques, autorisées à produire à partir de plans originaux. Ces fabriques n'ont pas été choisies au hasard : elles respectent l'environnement, les conditions de sécurité de leurs employés, et leur service juridique est capable de vérifier l'originalité des plans soumis.

Un inventeur apporte ses plans à l'une des fabriques, reçoit son produit manufacturé, et le vend. Un voyou achète le produit, l'étudie, et reproduit le plan à l'identique. Il fait à son tour produire l'objet, en bernant le service juridique d'une des fabriques, en menaçant ou en graissant la patte des bonnes personnes. Une fois l'objet fabriqué, pour ne pas se faire chahuter, le voyou vend ses contrefaçons à l'abri des regards. La nature des biens est donc inconnue des acheteurs, qui pensent acheter un produit original. Si et lorsque ces derniers apprennent la supercherie, le voyou est déjà loin et l'inventeur est floué.

Pour résoudre ce problème, il est décidé qu'une fabrique acceptant un plan qu'elle pense original doit contacter le tailleur de pierre, qui enregistrera l'événement sur son grand mur de marbre, réputé intangible. Ainsi, les inventeurs peuvent surveiller le mur, détecter qu'une fabrique a accepté un plan identique au sien et alerter la communauté de l'anormalité détectée. Il importe de noter que la fréquence des vérifications de l'inventeur est indifférente. Bien sûr, vérifier très fréquemment lui permet de détecter au plus tôt l'événement, mais cela importe peu : le méfait a été commis. Le système n'est cependant pas inutile : il vise à mettre la pression sur les fabriques. Elles sont désormais publiquement comptables de leurs exactions et peuvent voir leur réputation entachée, avec le risque de perdre la confiance de la communauté.

Plusieurs problèmes subsistent cependant :

- ⇨ il est nécessaire de faire confiance au tailleur de pierre ;
- ⇨ les fabriques ne sont pas vraiment contraintes de faire les déclarations à ce dernier.

Pour les résoudre, la communauté exige qu'un contrat soit signé par la fabrique et le tailleur de pierre pour chaque événement gravé sur le mur. Le contrat indique notamment que le tailleur s'engage à inscrire sur le mur les informations sous un court délai. Ce contrat permet l'émission de certificats d'authenticité, remis à chaque acheteur, et sur lesquels est noté le numéro de contrat.

Le contrat protège la fabrique contre un tailleur de pierre qui inscrirait n'importe quoi en vue d'entacher arbitrairement sa réputation. Par ailleurs, le contrat prévient les inscriptions illégitimes qui surviendraient lorsque le mur est laissé sans surveillance, puisque le tailleur de pierre doit donner son aval. Finalement, sans contrat, il est impossible d'émettre les certificats d'authenticité demandés par les acheteurs puisque le numéro de contrat y est représenté.

Il convient de noter que les acheteurs n'ont pas besoin de systématiquement vérifier le mur. Cela les obligerait à se rendre à l'autre bout du village et à perdre du temps. Le certificat d'authenticité permet d'accorder le bénéfice du doute et d'avoir une preuve d'un faux, ultérieurement.

En implémentant ces mesures de sécurité, le système de réputation est renforcé :

- ⇨ les fabriques fautives sont détectables et rendues comptables ;
- ⇨ les fautes sont prouvables par des documents non répudiables.

Il est désormais temps de considérer ce système sous l'angle technique !

2.2 Certificate Tranparency, d'un point de vue cryptographique

2.2.1 Clarification de la métaphore

Pour remettre en termes techniques la métaphore précédente, le lecteur est invité à considérer que les fabriques sont, en fait, les AC et les acheteurs sont les internautes. CT, tel que défini dans la RFC 6962 **[RFCCT]**, concentre ses efforts sur l'amélioration de la sécurité du Web, à l'exclusion des usages d'infrastructure de gestion de clés (IGC) ; les inventeurs sont donc des administrateurs de sites web. Finalement, le tailleur de pierre est le nouveau rôle introduit par CT, et jusque-là inconnu des IGC. Il s'agit du mainteneur d'un journal (*log*) cryptographique. Les journaux de CT ont la particularité d'être inviolables. En effet, une fois une donnée inscrite dans ces journaux, il est cryptographiquement impossible de l'en retirer ou de la remplacer, sans devoir tout réécrire à la suite de l'entrée altérée. Ces journaux étant publics, une telle réécriture pourra être détectée par des tiers surveillant ce genre d'événements. L'inviolabilité est donc atteinte par l'existence de preuves non répudiables d'altération.

2.2.2 Introduction aux arbres de Merkle

Pour implémenter ces journaux, CT utilise une structure de stockage appelée « arbre de Merkle », inventée en 1979 **[MerkleTree]**. Il s'agit d'une manière d'utiliser des fonctions de hachage cryptographiques, telles que SHA-2, afin de construire des arbres, comme celui illustré dans la figure 1. En fait, le lecteur les utilise probablement déjà quotidiennement sans le savoir, par l'entremise de Git ou de la *blockchain* de Bitcoin.

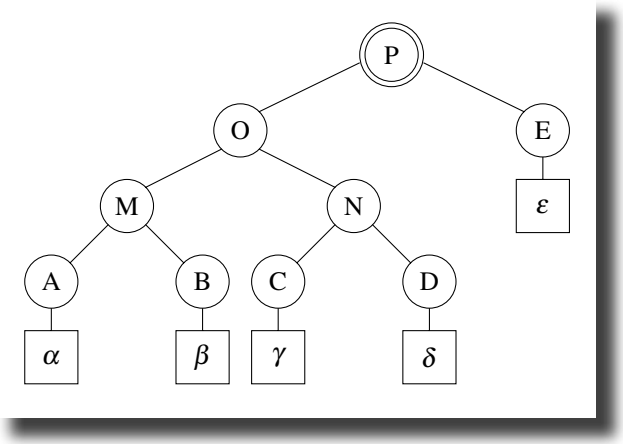


Fig. 1 : Un arbre de Merkle comprenant 5 éléments/feuilles.

Dans la figure 1, les données insérées dans l'arbre sont les feuilles de l'arbre : les nœuds α, β, γ, δ et ε. Ces feuilles sont hachées par la fonction H, par exemple SHA-256, et le résultat est noté **A**, **B**,..., **E**. Le nœud **M** est ensuite obtenu par la formule **M** = H(**A**||**B**)) avec || l'opérateur de concaténation. Il est fait de même pour former le nœud **N**. Le nœud **O** est ensuite calculé tel que **O** = H(**M**||**N**). Ainsi, **O** = H(H(H(α)||H(β))||H(H(γ)||H(δ))). Enfin, le nœud racine **P** est calculé, tel que **P** = H(**O**||**E**).

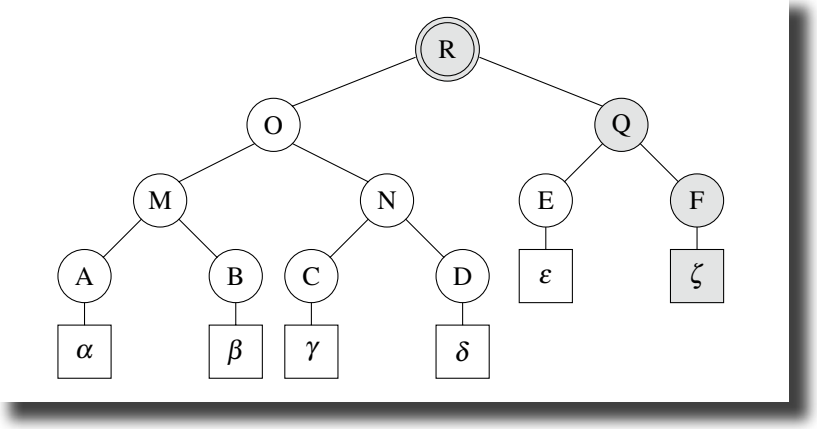


Fig. 2 : L'arbre de Merkle a 5 éléments/feuilles auxquels un sixième est ajouté. Les éléments altérés ou nouveaux sont grisés.

Pour ajouter une nouvelle feuille à cet arbre, notée ζ dans la figure 2, cette dernière est hachée pour générer **F**. Un nouveau nœud **Q** est créé tel que **Q** = H(**E**||**F**). Une nouvelle racine, notée **R**, remplace alors **P**, tel que **R** = H(**O**||**Q**).

Dans le cas de CT, chaque feuille de cet arbre contient un certificat X.509 accompagné de métadonnées. Elles peuvent aussi contenir, à la place, un pré-certificat, pour des raisons techniques que le lecteur intéressé pourra considérer en lisant la RFC. Pour continuer le parallèle, dans Git, les feuilles sont des *blob*, c'est-à-dire des fichiers, les nœuds intermédiaires sont les répertoires, et les racines sont les *commits*. Pour Bitcoin, les transactions sont les feuilles, et les blocs forment les racines.

Dans CT, pour assurer l'intégrité de l'arbre et pour engager la responsabilité du mainteneur du journal, les racines de l'arbre, c'est-à-dire les nœuds **P** et **R** dans les exemples précédents, sont signées cryptographiquement. Elles sont représentées par un double cercle dans les figures de cet article. Dans le jargon de CT, elles sont appelées *Signed Tree Head* (STH). La signature est générée avec des algorithmes à clé publique : soit RSA, soit ECDSA sur P-256. Cette signature est analogue à l'acte de gravure sur le mur de marbre par notre tailleur de pierre. Dans Git, cela revient à signer un *tag*.

2.2.3 Inscription de certificats dans l'arbre de Merkle

Un certificat protégé par CT doit être soumis en premier lieu à un ou plusieurs journaux par l'AC émettrice. Il y a au jour de l'écriture de cet article dix journaux référencés **[ListeLogs]**. Ce certificat n'est pas immédiatement inscrit au journal : il s'agit une opération qu'il est préférable de planifier régulièrement plutôt que de l'exécuter en temps réel. Certaines catégories d'attaques sont ainsi contrées, tout en régulant la mise en œuvre de la clé privée utilisée pour faire les signatures. Le nombre de preuves de cohérence, introduites ultérieurement dans cet article, est également restreint. Le journal délivre cependant, en réponse à cette soumission, une promesse d'insertion, datée et signée. Dans le jargon de CT, il s'agit d'un *Signed Certificate Timestamp* (SCT). La promesse doit être honorée avant un délai contractuel, appelé *Maximum Merge Delay* (MMD). Le certificat soumis d'une part, et le SCT d'autre part étaient symbolisés par le contrat signé en deux exemplaires par le tailleur de pierre et la fabrique dans l'illustration précédente.

2.2.4 De l'usage des SCT

Le SCT est fourni aux navigateurs à chaque fois qu'ils établissent une connexion TLS. Son rôle est équivalent à celui du certificat d'authenticité dans l'illustration précédente. Le SCT est transmis, au choix :

- ⇨ au travers d'une extension TLS spécifique (type 18). La figure 3 montre cette extension grâce à une capture réseau analysée avec l'outil Parsifal **[Parsifal]**. Elle est prise en charge par Nginx 1.9.0 ou ultérieur, et Apache et HAProxy dans leurs versions en cours de développement :

Terminal

```
$ parsifal --pcap-tcp 443 -T tls --always-enrich -g '**.server_
extensions.*.extension_type' /tmp/serverhello.pcap
[[RenegotiationInfo, ServerName, Unknown extension_type (23),
SessionTicket, Unknown extension_type (18), Unknown extension_type (16),
Unknown extension_type (30032), ECPointFormats]]
$ parsifal --pcap-tcp 443 -T tls --always-enrich -g '**.server_extensions.
[4].extension_data' /tmp/serverhello.pcap
[[Unparsed]_00f0007600ee4bbdb775ce60bae142691fabe19e66a30f...
```

Fig. 3 : Capture réseau d'une connexion à google.com illustrant une négociation TLS avec l'extension 18, afin de publier un SCT dans le SERVER_HELLO.

⇨ dans un message de contrôle de révocation agrafé (*OCSP stapling*) ;

⇨ dans le certificat, le SCT étant stocké dans une extension X.509v3. La figure 4 montre l'extraction d'une telle extension avec l'outil Scapy **[SCAPY]**.

Fichier

```
from scapy.all import *
cert_raw = open('particuliers.societegenerale.fr.der').read()
cert = X509_Cert(c)
for ext in cert.tbsCertificate.extensions:
  if ext.extnID.val == '1.3.6.1.4.1.11129.2.4.2':
    ext.show()
```

Terminal

```
###[ X509_Extension ]###
  extnID= <ASN1_OID['.1.3.6.1.4.1.11129.2.4.2']>
  critical= None
  \extnValue\
   |###[ X509_ExtDefault ]###
   |  value= <ASN1_STRING['\x01g ...
```

Fig. 4 : Code Scapy permettant l'affichage hexadécimal de l'extension X.509v3 contenant le SCT dans l'OID 1.3.6.1.4.1.11129.2.4.2.

Un SCT est exigé par le navigateur Chrome pour tous les certificats X.509 de qualité *Extended Validation* (EV) émis depuis le 1er janvier 2015. Pour mémoire, les certificats EV sont ceux émis après les contrôles d'identité des demandeurs les plus stricts. Sans SCT, ces certificats sont désormais considérés invalides.

En outre, les SCT sont conservés par les navigateurs au-delà du processus de vérification des certificats qu'ils accompagnent. Ces derniers sont en effet utilisés régulièrement pour demander des preuves d'insertion, appelées *Merkle Inclusion Proof*. Ces preuves ont pour objectif de vérifier que la promesse d'insertion a été honorée passé le MMD. Détecter qu'un SCT n'est pas dans le journal passé ce délai est une preuve de trahison du journal qui tente de masquer qu'un certificat frauduleux a été émis.

NOTE

Une preuve d'insertion est composée du sous-ensemble minimal de nœuds d'un arbre de Merkle nécessaire pour rattacher une feuille à la racine. Les nœuds D, M, et Q forment la preuve d'insertion de γ, pour peu que le R' calculé soit égal au R de la figure 2.

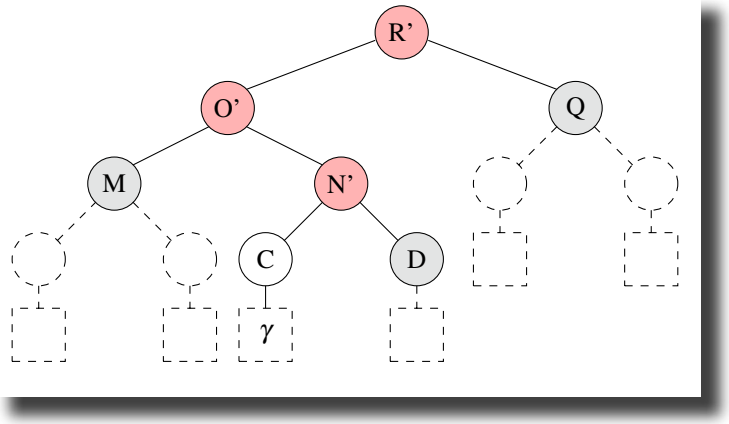


Fig. 5 : Illustration d'une preuve d'insertion, représentée par les nœuds gris. Les nœuds roses sont calculés à partir de la preuve et de γ pour former R'.

2.2.5 Surveiller les journaux

CT nécessite que les administrateurs de sites web vérifient qu'aucune AC n'a émis de certificat frauduleux. Pour cela, ils doivent fréquemment consulter les journaux et détecter les émissions de certificats qu'ils n'ont pas sollicitées. S'ils ne le font pas, CT ne sert strictement à rien.

Enfin, il est nécessaire de vérifier l'intégrité des journaux, en exigeant des preuves de cohérence, appelées, dans le jargon, *Merkle Consistency Proofs*. Ces preuves sont sollicitées par des acteurs, nommés moniteurs. Leur rôle est de vérifier qu'une racine signée, c'est-à-dire un STH, est bien l'ancêtre d'une racine signée plus récente. Pour reprendre les figures 1 et 2, cela veut dire qu'un moniteur doit réussir à valider qu'il est possible de faire évoluer l'arbre ayant pour racine **P** jusqu'à obtenir l'arbre ayant pour racine **R**, en rajoutant des éléments un à un. Ce faisant, il est possible de prouver que le journal n'a pas été réécrit pour dissimuler des certificats après qu'une attaque a été effectuée. En effet, en cas de réécriture, il existerait un STH qu'il serait impossible de faire évoluer vers le STH le plus récent.

Google fournit plusieurs implémentations pour effectuer ces vérifications **[RefImpl]**. Bien qu'il soit présagé que la plupart des moniteurs soient des ACs, toute personne peut les lancer afin de décentraliser la vérification.

3. PARTITIONNEMENT DE LA VUE, ET JOURNAUX MULTIPLES

3.1 Scénario d'attaque par un journal malhonnête

À ce jour, CT est encore incomplet : la confiance n'a été que déplacée des AC aux journaux. Dans la théorie, les journaux sont consultables et vérifiables par tous. Néanmoins, encore faut-il que tout le monde accède aux mêmes journaux !

Lorsqu'un journal malhonnête sert une donnée différente en fonction de la personne qui l'interroge, il effectue une attaque par « partitionnement de la vue ». Ce type d'attaques n'est pas spécifique à CT et est bien connu, par exemple des experts DNS.

Le principe général d'une telle attaque est le suivant. La victime visite un site web pour lequel l'attaquant effectue un *Man-in-the-Middle*. La victime reçoit alors un certificat frauduleux qu'il considère valide, car il a été émis par une autorité de certification réputée « de confiance ». Avec ce certificat, il reçoit aussi un SCT, qu'il vérifie, lui prouvant ainsi que le mainteneur du journal s'est bien engagé à inscrire le certificat reçu pendant l'attaque. À ce stade, la connexion est alors bien interceptée par l'attaquant et la victime en subit tous les désagréments. C'est le fonctionnement normal de CT ! Il est en effet important de se souvenir que CT ne prévient pas les attaques ; il permet juste de les détecter et de pouvoir les prouver *a posteriori*.

Plus tard, un processus d'audit se déclenche ; le navigateur de la victime sélectionne alors à son gré des SCT qu'il a collectés durant la navigation. Parmi ceux-ci peut figurer le SCT reçu pendant l'attaque.

Pour chaque SCT audité, une preuve d'insertion (*Inclusion Proofs*) du certificat associé au SCT doit être fournie par le journal. L'incapacité de fournir une preuve démontre une déficience technique ou délibérée du journal à insérer le certificat avant l'expiration du MMD. Il s'agit d'un incident de sécurité impliquant la responsabilité du journal ; il est, en effet, impossible pour un tiers de déterminer si l'absence du certificat est une attaque visant à « dissimuler » un certificat frauduleux ou non.

La fourniture d'une preuve d'insertion valide ne garantit cependant rien de l'honnêteté éventuelle d'un journal. En effet, le mainteneur d'un journal malhonnête peut créer un second journal, dérivé du premier et construit à l'unique attention de la victime. Dans cet autre journal, le certificat utilisé pour l'attaque est bien inséré et une preuve de son insertion peut donc être délivrée. En revanche, dans le « vrai » journal, présenté à tous les autres utilisateurs de CT, le certificat frauduleux n'est jamais inséré. Ce partitionnement de la vue empêche le propriétaire du site intercepté de voir le certificat frauduleux dans le « vrai » journal. D'autre part, la victime de l'attaque *Man-in-the-Middle* ne peut se douter de rien, ayant validé avec succès toutes les preuves à sa disposition.

3.2 Une contre-mesure

Pour contrecarrer le partitionnement de vue, il est possible de faire circuler sur Internet les SCT reçus. Un SCT pour un certificat inséré seulement dans un « faux » journal finira ainsi par être audité par un utilisateur à qui le journal ne mentira pas ou ne pourra pas mentir sans dévoiler la supercherie. Ce SCT constituera alors une preuve non répudiable de la trahison du journal.

L'IETF travaille sur des moyens de diffusion des SCT et des STH par des mécanismes de rumeurs (*gossip*) utilisant des points d'échanges ou des communications *peer-to-peer*. Ces méthodes n'étant pas encore complètement normalisées **[I-D]**, leur description ne sera pas détaillée dans cet article. Il peut être néanmoins intéressant de noter que l'un des défis à relever est un problème de vie privée : communiquer un SCT signifie qu'un utilisateur a visité le site qui le lui a remis en premier lieu.

En attendant, CT sert principalement à détecter les émissions de certificats accidentelles ou effectuées à la suite d'une compromission ou de pressions sur les AC.

En l'absence de ces rumeurs, aucune garantie n'existe contre un attaquant agissant simultanément sur une AC et un journal.

4. QUEL AVENIR POUR CT ?

CT est encore jeune. Il est néanmoins probable qu'il soit largement adopté. Il est, en effet, promu par quelques géants de l'Internet, dont Google, à l'origine de cette technologie, et développeur de Chrome, le navigateur le plus utilisé de nos jours.

Google ne se cache pas de faire pression sur les AC, prenant depuis plusieurs années, des mesures contre les AC fautives en leur imposant la prise en charge de CT. Ainsi, dans l'affaire récente ayant impliqué l'émission de certificats frauduleux « de tests » par Symantec, Google a exigé que tous les certificats de cette AC soient inscrits dans CT, quelle que soit la qualité du certificat : EV et non-EV **[Symantec]**. Il est probable que cette démarche se généralise progressivement. La nouvelle AC *Let's Encrypt* a, d'ailleurs, montré la voie, en se soumettant à CT de son propre chef **[LetsCT]**. Des initiatives similaires de la part des autres AC seraient souhaitables.

D'un point de vue technique, CT a besoin de parfaire son système de rumeurs afin de contrer les attaques par partitionnement de vue de journaux malhonnêtes. Il est aussi nécessaire de réfléchir à un moyen de limiter la taille des journaux. Il faudra alors trouver une façon satisfaisante de défausser les entrées expirées tout en maintenant la confiance. Enfin, le risque de déni de service sur les journaux **[DDoS]** est une problématique croissante pouvant affecter la confiance en les journaux qui en sont victimes : sont-ils vraiment indisponibles ou cachent-ils quelque chose ?

Par ailleurs, le principe de CT est générique et peut être décliné pour d'autres technologies ayant des modèles de confiance et de réputation à renforcer. Ainsi, le groupe de travail **trans [WG]** de l'IETF travaille à adapter cette technologie pour DNSSEC.

CONCLUSION

Compte tenu des limitations et des objectifs de CT, le lecteur peut légitimement s'interroger sur la capacité de CT à améliorer la situation.

Certes, CT n'est pas parfait. Néanmoins, il offre une approche pragmatique et déployable rapidement. Ainsi, depuis début 2015, CT permet à tous de surveiller l'émission de certificats frauduleux et de prouver publiquement des exactions qui n'étaient pas toujours détectées auparavant. Il s'agit donc d'un excellent outil permettant de renforcer le système de réputation par la détection des méfaits grâce à de nouvelles preuves cryptographiques non répudiables. ▮

REMERCIEMENTS

Un grand merci à mes relecteurs pour leur patience, les commentaires bienveillants et leurs apports significatifs : D. Diallo, O. Levillain, J. Rossi, M. Tury, G. Valadon et N. Vivet.

RÉFÉRENCES

⇨ [Dunbar] **https://goo.gl/Fe9YsH** (Wikipedia EN)

⇨ [poule] **https://goo.gl/bUqeQr** (Wikipedia FR)

⇨ [Schneier] ISBN-13: 978-1118143308

⇨ [CABFORUM] **https://cabforum.org/**

⇨ [CertLimit] **https://goo.gl/2oXDQ5** (Google Security Blog)

⇨ [EFFObs] **https://www.eff.org/observatory/**

⇨ [ChromeSurv] **https://googleonlinesecurity.blogspot.fr/2013/01/enhancing-digital-certificate-security.html**

⇨ [CNNIC] **https://googleonlinesecurity.blogspot.fr/2015/03/maintaining-digital-certificate-security.html**

⇨ [IndiaNIC] **https://googleonlinesecurity.blogspot.fr/2014/07/maintaining-digital-certificate-security.html**

⇨ [Diginotar] **https://googleonlinesecurity.blogspot.fr/2011/08/update-on-attempted-man-in-middle.html**

⇨ [Perspectives] **http://perspectives-project.org/**

⇨ [RFCCT] **http://www.rfc-editor.org/rfc/rfc6962.txt**

⇨ [MerkleTree] **http://goo.gl/QMuSfa** (Springer – PDF (735KB))

⇨ [RefImpl] **https://github.com/google/certificate-transparency**

⇨ [ListeLogs] **https://www.certificate-transparency.org/known-logs**

⇨ [Parsifal] **https://github.com/ANSSI-FR/parsifal**

⇨ [SCAPY] **https://bitbucket.org/secdev/scapy/overview**

⇨ [Symantec] **https://goo.gl/uHWV0k** (Google Security Blog)

⇨ [LetsCT] **https://letsencrypt.org/certificates/**

⇨ [DDoS] **http://goo.gl/6A04bg** (Chromium's Certificate Transparency policy mailing list)

⇨ [WG] **https://datatracker.ietf.org/wg/trans/documents/**

LETSENCRYPT : HTTPS ET TLS ENFIN INDUSTRIALISÉS

Benjamin SONNTAG

En novembre 2014, le projet Letsencrypt, lancé par Mozilla, Cisco, Akamai, EFF et Identrust, permet désormais d'héberger des sites en HTTPS presque aussi facilement qu'en HTTP. Dans cet article, nous allons vous raconter l'histoire de Letsencrypt, son fonctionnement, ses limites, et l'avenir qu'il promet à l'industrie de l'hébergement.

1. UNE PETITE HISTOIRE DE HTTPS

Letsencrypt, un projet américain de la société civile et de grands constructeurs de logiciels et de matériels, vise à améliorer la sécurité d'Internet en rendant le chiffrement plus répandu.

En 1995, le Web était en clair : toutes les requêtes HTTP passaient non chiffrées sur Internet. Lorsque Netscape Navigator introduit SSL cette année-là, une première révolution a lieu sur le net : le chiffrement des échanges de pages web devient possible, permettant le développement du commerce en ligne, la banque par Internet et l'accès sécurisé aux espaces privés d'entreprises.

Avec l'arrivée d'HTTPS, donc de SSL puis TLS, est aussi apparu le principe des certificats X.509 vendus aux propriétaires de sites web. Ces fichiers permettent de créer une chaîne de confiance entre une autorité de certification, société dont la clé publique est fournie nativement dans les navigateurs, et le propriétaire d'un site web, dont la clé publique du serveur TLS est ainsi reconnue comme valide pour un nom de domaine donné. Depuis 2005, l'ensemble des règles qui régissent les autorités de certification et leur inclusion dans les navigateurs est décidé par le CA/Browser Forum, un consortium regroupant les CA et constructeurs de navigateurs **[1]**.

En 1995, les certificats SSL/TLS X.509 étaient délivrés au compte-goutte, après paiement de sommes importantes à l'autorité de certification chargée de valider la propriété du nom de domaine auprès du webmaster, le tout étant réalisé manuellement. Peu à peu, l'automatisation a gagné ce métier, donnant une distinction entre les certificats DV (*Domain Validation*) et EV (*Extended Validation*). Le premier (DV) pouvant être automatisé, il ne valide que le fait que vous possédez un nom de domaine, donc que vous pouvez publier une page sur ce dernier, ou un enregistrement particulier dans le DNS. Le second (EV) nécessite de prouver l'existence d'une personne physique ou morale propriétaire du domaine, et affiche une barre verte à votre nom dans les navigateurs.

Pour les certificats DV, une automatisation étant donc possible, et de nombreux fournisseurs comme Comodo ou Gandi permettaient d'utiliser une API pour obtenir des certificats X.509 reconnus. Cependant, le coût, bien que nettement inférieur aux 1000$ des débuts, restait non nul (en 2015, de 7 à 12€HT pour un certificat mono-domaine valable un an), ou n'était pas automatisé : StartCom, par exemple, fournissait des certificats gratuits pour les individus, au prix d'une paperasse importante et interdit aux professionnels. Cela bloquait donc l'adoption en masse de TLS sur les serveurs web ou pour d'autres protocoles (SMTP, IMAP, XMPP etc. savent aussi faire du TLS, et nécessitent donc un certificat pour valider leur nom).

2. NAISSANCE DE LETSENCRYPT

En 2010, l'EFF, l'*Electronic Frountier Foundation*, association américaine de défense des libertés, publia un greffon pour Firefox et Chrome, nommé HTTPS Everywhere **[2]**, qui permet de s'assurer que l'on utilise bien uniquement HTTPS sur les sites qui gèrent ce protocole correctement. Cependant, peu de sites étant concernés, ce greffon n'avait d'utilité que pour les principaux gros silos : Facebook, Google, Amazon & consorts. À ce jour, il référence environ 20 000 sites web.

En 2012, suite aux révélations d'Edward Snowden sur la surveillance de masse des communications des citoyens par l'agence de renseignement américaine NSA, l'EFF souhaita passer la vitesse supérieure en proposant des certificats TLS gratuits et accessibles facilement à tous, via une API, le financement de ce projet devant être assuré par quelques industriels et sponsors. Le projet « Letsencrypt » est donc né courant 2014.

Pour cela, Mozilla, l'EFF et Identrust, ainsi que quelques sponsors (Cisco, Akamai, Facebook...) ont lancé un projet de RFC pour un protocole d'obtention automatique de certificats via une API : le protocole ACME (pour *Automatic Certificate Management Environment*), et démarré 2 logiciels libres : Boulder, le serveur ACME d'autorité de certification, écrit en Go, et Letsencrypt, le client ACME d'enregistrement et de révocation de certificats X.509. Ce dernier ayant reçu des contributions du monde entier **[3]**.

3. FONCTIONNEMENT DE LETSENCRYPT

3.1 Installation et obtention d'un premier certificat

Pour utiliser Letsencrypt, le plus simple est d'utiliser leur client natif, disponible sur leur compte GitHub : **https://github.com/letsencrypt/letsencrypt** et de suivre le mode d'emploi de leur site à l'adresse **https://letsencrypt.org/howitworks/**.

Sur une distribution Linux, on clone leur dépôt, et on lance une première fois letsencrypt (en tant que root) : il installe alors les dépendances requises et rappelle les options de la ligne de commandes :

Terminal

```
server:~# git clone https://github.com/letsencrypt/letsencrypt
Cloning into 'letsencrypt'...
server:~# cd letsencrypt/
server:~/letsencrypt# ./letsencrypt-auto  --help
Updating letsencrypt and virtual environment dependencies......
Requesting root privileges to run with virtualenv: /root/.local/share/
letsencrypt/bin/letsencrypt --help

  letsencrypt [SUBCOMMAND] [options] [-d domain] [-d domain] ...

The Let's Encrypt agent can obtain and install HTTPS/TLS/SSL certificates. By
default, it will attempt to use a webserver both for obtaining and installing
the cert. Major SUBCOMMANDS are:
  (default) run        Obtain & install a cert in your current webserver
  certonly             Obtain cert, but do not install it (aka "auth")
  ...
Choice of server plugins for obtaining and installing cert:

  --apache           Use the Apache plugin for authentication & installation
  --standalone       Run a standalone webserver for authentication
  (nginx support is experimental, buggy, and not installed by default)
  --webroot          Place files in a server's webroot folder for
authentication
  ...
```

Letsencrypt souhaitant faciliter au maximum l'obtention et la configuration d'un serveur web en HTTPS, il dispose de plugins pour valider la propriété du nom de domaine via HTTP, soit via le serveur web Apache (option `--apache`), soit en créant son propre serveur web python minimaliste le temps de la validation (`--standalone`), soit en expliquant à letsencrypt où se trouve la racine web (`--webroot`).

Une façon simple d'obtenir un certificat sans laisser letsencrypt toucher à votre serveur web consiste, sur un serveur Apache ou Nginx existant, à disposer d'un alias HTTP pour le motif d'URL `/.well-known/acme-challenge` (il s'agit de l'URL où boulder va valider la propriété du nom de domaine) sur un dossier local. Par exemple, avec Apache on configurera un hôte virtuel ainsi (ce n'est qu'un exemple, il se peut que vous soyez obligé d'utiliser une ligne RewriteRule à la place) :

Terminal

```
server:~# mkdir -p /var/www/letsencrypt/.well-known/acme-challenge
server:~# cat /etc/apache2/sites-enabled/default
  <VirtualHost *:80>
  ...
      Alias /.well-known/acme-challenge /var/www/letsencrypt/.well-known
/acme-challenge
  </VirtualHost>
```

Avec Nginx, on pourra utiliser un bloc location/alias comme suit :

Terminal

```
location /.well-known/acme-challenge {
    alias /var/www/letsencrypt/.well-known/acme-challenge;
}
```

Ensuite, on utilisera l'option `--webroot` et sa sous-option `-w <webroot>` pour obtenir un certificat facilement :

Terminal

```
server:~# ./letsencrypt-auto certonly --webroot -w /var/www/letsencrypt -d
www.example.com
```

Au premier lancement, Letsencrypt vous demande une adresse e-mail (pour vous prévenir en cas de problème) ainsi que l'acceptation de leurs conditions générales. Cette étape vous crée un compte sur le site de letsencrypt, associé à une clé privée RSA ou ECDSA stockée sur votre serveur.

Une fois cette étape d'inscription passée, le client Letsencrypt en python valide la propriété de votre nom de domaine et demande un certificat pour ce dernier :

Terminal

```
Requesting root privileges to run with virtualenv: /root/.local/share/
letsencrypt/bin/letsencrypt certonly --webroot -w /var/www/letsencrypt -d
www.example.com

IMPORTANT NOTES:
 - Congratulations! Your certificate and chain have been saved at
   /etc/letsencrypt/live/www.example.com/fullchain.pem. Your cert will
   expire on 2016-04-18. To obtain a new version of the certificate in
   the future, simply run Let's Encrypt again.
```

Les certificats et clés privées Letsencrypt sont stockés par défaut dans `/etc/letsencrypt/archive/<nom de domaine>/cert1.pem` avec un lien symbolique vers le dernier certificat obtenu pour un domaine donné dans `/etc/letsencrypt/live/<nom de domaine>/cert.pem`.

Pourquoi « vers le dernier certificat obtenu » ? Tout simplement parce que comme tous les certificats d'autorité de certification, ceux-ci ont une durée de vie déterminée. Les ingénieurs derrière Letsencrypt ont fait le choix de **produire des certificats valables 90 jours** seulement, contrairement aux 1 à 3 ans des autorités habituelles. On verra que ce choix, qui certes pousse très fort les administrateurs de serveurs à automatiser le processus d'obtention des certificats, n'est pas sans conséquence.

3.2 Renouvellement

Les certificats de Letsencrypt n'étant valable que 90 jours, ils recommandent de les renouveler au bout de 60 jours d'utilisation (soit 30 jours avant l'expiration). À ce jour, le programme letsencrypt ne sait renouveler les certificats que via la ligne de commandes en mode interactif (via un masque de saisie demandant à l'utilisateur de confirmer son action) et uniquement dans le cadre de l'utilisation du greffon apache.

Conséquence de cela, chacun se débrouille pour l'instant pour mettre à jour automatiquement ses certificats. Dans le cadre d'un serveur hébergeant de nombreux certificats, on peut utiliser un script comme celui ci-dessous, que l'on lancerait quotidiennement via une tâche planifiée `cron` par exemple. Ce script requiert `bash`, `sed`, `openssl` et bien entendu ... letsencrypt [4].

Terminal

```
#!/bin/bash

# Send alert mail to this address:
ALERT=sslalert@octopuce.fr
LETSENCRYPT_BIN=/usr/local/letsencrypt/letsencrypt-auto
WEBROOT=/var/www/letsencrypt

cd "$(dirname $0)"
HERE="$PWD"
DATE_TODAY=$(date +'%s')
cd /etc/letsencrypt/live || (echo "Can't cd to /etc/letsencrypt/live !" &&
exit -1)
for domain in *
do
    if [ -f "$domain/cert.pem" ] ; then
 CERT="$domain/cert.pem"
 CERT_END_DATE=$(openssl x509 -in "$CERT" -noout -enddate | sed -e
"s/.*=//")
 DATE_CERT=$(date -ud "$CERT_END_DATE" +"%s")
 DATE_JOURS_DIFF=$(( ( $DATE_CERT - $DATE_TODAY ) / (60*60*24) ))
 if [[ $DATE_JOURS_DIFF -le 30 ]]; then
     echo "Trying to renew certificate for domain $domain expiring in
$DATE_JOURS_DIFF days"
     # Read the SAN (Subject Alt Names) for this cert (Warn: this code may
not be super reliable :/ )
     SAN=$(openssl x509 -in "$CERT"  -text|grep DNS:|sed -e "s/DNS:/-d /g"
-e "s/, / /g")
     # Try to renew it:
     $LETSENCRYPT_BIN certonly --webroot -w "$WEBROOT" -d "$domain" $SAN
     if [ "$?" -ne "0" ]
     then
   echo "Certificate /etc/letsencrypt/live/$domain has NOT been
successfully renewed, please check" | mail -s "Can't renew certificate
$domain on $(hostname)" $ALERT
     fi
 fi
    fi
done
```

On pourra y ajouter un rechargement des services concernés : le serveur web sûrement, et peut-être postfix, dovecot, (si votre mail utilise aussi TLS), prosody... Enfin, pour arrêter de renouveler un certificat particulier c'est simple : il suffira de supprimer le dossier `/etc/letsencrypt/live/<nom du domaine concerné>`.

3.3 Certificats complexes

Le type de certificats que l'on peut obtenir par Letsencrypt est simple : pas de certificats wildcard (permettant de gérer tous les sous-domaines de premier niveau d'un domaine donné) ni de certificats EV (validant l'identité de l'entreprise ou du particulier propriétaire). Par contre, il est possible d'obtenir des certificats multi-domaines. Pour cela, il suffit de faire pointer tous les FQDN sur le même serveur et de mettre plusieurs options `-d` à l'appel à letsencrypt :

Terminal

```
server:~# ./letsencrypt-auto certonly --webroot -w /var/www/letsencrypt -d
www.example.com -d dev.example.com -d www.monsite.fr
```

Dans ce cas, letsencrypt générera un certificat dont le nom principal est www.example.com, et disposant d'un champ `X509 Subject Alt Names` (SAN) reprenant tous les FQDN validés. Vous aurez sûrement remarqué que le script proposé en 3.2 gère ce cas via la variable `$SAN`.

On pourra aussi ajouter l'OCSP Stapling **[5]** à son serveur web, qui permet d'éviter que le navigateur doive vérifier la validité du certificat auprès de l'autorité lors des visites. On le configure ainsi :

⇨ pour Apache (>=2.3.3) :

Terminal

```
SSLCACertificateFile /etc/ssl/certs/ca-certificates.crt
SSLStaplingCache shmcb:/tmp/stapling_cache(128000)
SSLUseStapling on
```

⇨ pour Nginx (>=1.3) :

Terminal

```
ssl_stapling on;
ssl_stapling_verify on;
ssl_trusted_certificate /etc/ssl/certs/ca-certificates.crt ;
```

4. LIMITES ET DÉFAUTS DE LETSENCRYPT (ET DE L'ÉCOSYSTÈME TLS)

Maintenant que l'on est capable de jouer avec letsencrypt et d'obtenir un certificat pour passer son site en HTTPS, quels sont les limites et défauts du projet Letsencrypt à ce jour ?

Ils sont de plusieurs natures : tout d'abord, il y a les facteurs indépendants de letsencrypt pour passer un site web en HTTPS : Il ne suffit pas (et de loin) de disposer d'un certificat

et de passer le virtualhost de son serveur web en HTTPS, il faut bien configurer ce dernier pour fournir une sécurité décente, par exemple en validant la configuration sur le site SSLLabs de Qualys : **https://www.ssllabs.com/ssltest/** et essayer d'obtenir a minima un A-, probablement avec l'aide de la documentation (complet et en anglais) de Mozilla : **https://wiki.mozilla.org/Security/Server_Side_TLS**.

Ensuite, on n'oubliera pas de fournir le **certificat intermédiaire** avec le certificat du serveur, sans quoi un internaute arrivant pour la première fois sur votre site pourrait se voir afficher une erreur de validation (SSLLabs le signale en orange dans ce cas). Le programme de Letsencrypt résout ce problème en fournissant un fichier `fullchain.pem` dans le dossier `live/` qu'il faut utiliser au lieu de `cert.pem` dans votre configuration Apache ou Nginx comme suit :

⇨ Apache 2.2 & 2.4 :

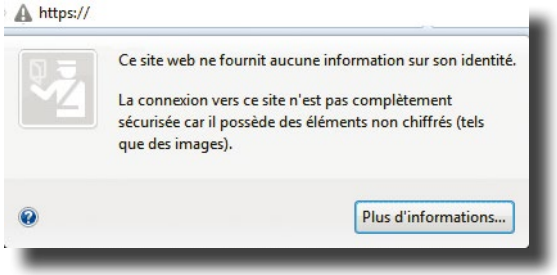
Terminal
```
SSLCertificateFile    /etc/letsencrypt/live/www.example.com/fullchain.pem
SSLCertificateKeyFile   /etc/letsencrypt/live/www.example.com/privkey.pem
```

⇨ Nginx :

Terminal
```
ssl_certificate     /etc/letsencrypt/live/www.example.com/fullchain.pem;
ssl_certificate_key /etc/letsencrypt/live/www.example.com/privkey.pem;
```

Enfin, lorsque l'on renouvelle un certificat, il ne faut pas oublier de recharger les logiciels l'utilisant pour qu'ils puissent prendre en compte le nouveau fichier. Le renouvellement n'étant pas totalement intégré au client python actuel, il faudra probablement le coder soi-même...

Un autre problème très classique est d'oublier qu'en 2016, un site web est souvent constitué de nombreuses **ressources externes** : polices de caractères, tracker pour les statistiques du site, images sur des CDN etc. Passer son site web en HTTPS requiert de fournir l'ensemble de ses sous-documents de page web en HTTPS sans quoi les navigateurs web refuseront de charger ces sous-éléments, les sachant non sécurisés. Dans ce cas, un cadenas brisé ou un bouclier apparaîtront sur les navigateurs.



Une fois HTTPS déployé sur un site, on doit souvent travailler l'ensemble du site pour éviter les contenus mixtes !

Enfin, la limitation à 90 jours de la durée des certificats de Letsencrypt peut représenter un problème pour certains protocoles associés à HTTPS.

4.1 HPKP, HSTS, DANE, comment interagissent-ils avec Letsencrypt ?

Le monde des autorités de certifications étant rempli de sociétés ayant été piratées (voir DigiNotar en Hollande [6]) ou ayant échoué à sécuriser leur validation ou leurs clés (voir Symantec, sérieusement bousculé par Google fin 2015 [7]), de nouvelles

normes sont apparues pour permettre d'améliorer la sécurité d'un site en HTTPS, à savoir de valider l'identité du site pour une clé donnée. Les certificats produits par Letsencrypt peuvent-ils être utilisés avec ces nouveaux protocoles dans de bonnes conditions ?

HSTS (pour *HTTP Strict Transport Security*) est une RFC (**https://tools.ietf.org/html/rfc6797**) qui décrit comment un serveur web peut signifier à un navigateur qu'il doit toujours utiliser HTTPS pour se connecter à un domaine donné : la première fois que vous arrivez sur un site, ce dernier vous signale ce fait avec un en-tête HTTP spécial : `Strict-Transport-Security`. Le navigateur se souvient alors de cela pendant une durée précisée (de quelques heures à quelques mois). Ainsi, si vous arrivez, plus tard, sur un réseau détournant le flux HTTP, votre navigateur ne sera pas perturbé lorsque vous irez sur ce site, puisqu'il se rendra automatiquement dessus en HTTPS même si vous ne le précisez pas.

Le risque suivant peut se poser avec Letsencrypt : si vous précisez via un entête HSTS qu'il faut rester en HTTPS sur votre site web, avec par exemple une mise en cache de 1 an, il faudra vous assurer de renouveler votre certificat tous les 60 à 90 jours, sans quoi HTTPS ne marchera plus, et vos clients arrivant automatiquement par HTTPS sur votre site, ne pourront plus du tout y accéder ! Ce n'est pas en soi un inconvénient de Letsencrypt, mais pensez-y lors de l'installation d'un en-tête HSTS.

HPKP est une autre RFC (**https://tools.ietf.org/html/rfc7469**), qui permettra de préciser qu'un site web dispose d'un certificat particulier ou est signé par une autorité de certification donnée. Pour cela, le serveur web fournit un en-tête spécial : `Public-Key-Pins` ou `Public-Key-Pins-Report-Only` qui précise l'empreinte cryptographique du certificat du serveur ou de celui d'une autorité de certification racine ou intermédiaire qui l'a signé. Par exemple :

Terminal

```
Public-Key-Pins: max-age=31556736; pin-sha256="d6qzRu9zOECb90Uez27xWltN
sj0e1Md7GkYYkVoZWmM=";
```

Si l'on fournit l'empreinte d'un certificat serveur fournit par Letsencrypt, on voit rapidement le problème : ce certificat expirant dans 90 jours, il faudrait par exemple limiter le max-age à 60 jours, puis 30, puis 0, et effectuer un key-rollover : annoncer en avance la nouvelle empreinte en plus de l'actuelle, avant de remplacer le certificat avec le nouveau. La complexité d'un tel processus, bien connu de ceux qui déploient du DNSSEC, le rend inexploitable.

Aussi, on préférera publier dans l'en-tête `Public-Key-Pins` l'empreinte du certificat intermédiaire de Letsencrypt lui-même, comme le permet le protocole HPKP, plutôt que celui du certificat final. Aussi, tant que votre site web sera signé par l'autorité de Letsencrypt, vous n'aurez pas à changer votre en-tête HPKP, cela vous protège donc de la création de certificats illégaux sur d'autres autorités, soit par piratage, soit par enjeu politique (la Turquie, la Chine, la France, et d'autres pays, ont des CA nationales susceptibles de produire des certificats en dehors des règles du CAB Forum, même si cela leur fait risquer leur révocation des navigateurs). Notez que cela ne vous protégera pas si Letsencrypt crée un certificat frauduleux pour votre site web : le navigateur verra ce dernier comme valide.

On notera que HPKP étant très récent, il n'est pas, à ce jour, utilisé par les principaux navigateurs du commerce, et cette inconvénient ne l'est donc que pour les plus geeks d'entre nous qui utiliseront des greffons de navigateur validant l'empreinte des certificats fournis via HPKP.

DANE/TLSA est enfin une dernière RFC, qui propose de mettre l'empreinte de votre certificat TLS ou de sa clé publique dans le DNS plutôt que sous forme d'en-tête HTTP. Tout d'abord cela a l'avantage de permettre de fournir par un autre protocole l'empreinte de la clé publique d'un serveur, donc de diversifier les canaux de distribution. Ce faisant, DANE/TLSA protège donc aussi les protocoles comme SMTP, IMAP ou XMPP utilisant TLS. Enfin, si la zone de votre domaine est signée par **DNSSEC**, votre empreinte de certificat est donc fournie avec une validation cryptographique forte, du même niveau que les autorités de certification.

Avec DANE et la ressource DNS de type TLSA, le même problème se pose cependant qu'avec HPKP : si vous publiez l'empreinte de votre certificat final Letsencrypt, il faudra procéder à une rotation de cette empreinte tous les 60 jours, ce qui est une tâche complexe et risquée, sauf à l'automatiser.

Letsencrypt vous recommande donc **[8]**, dans TLSA, soit de publier non pas l'empreinte de votre certificat, mais celle de votre clé publique, et de conserver la même clé publique un certain temps (par exemple un ou deux ans), soit de publier l'empreinte de l'autorité de certification présente dans les navigateurs, et donc de la fournir dans votre serveur web en plus de celle de letsencrypt, comme le précise la RFC de DANE.

Un exemple d'enregistrement DNS TLSA serait :

Terminal

```
_25._tcp.z1.octopuce.fr IN TLSA 3 0 1 (
b8828d9953fbdcfbfc6d0680eb0eb74e985ff361045505db627edc5442d18345 )
```

Cet enregistrement précise l'empreinte du certificat TLS du serveur SMTP situé à l'adresse **z1.octopuce.fr**.

En conclusion, on constate que la durée limitée (90 jours) des certificats de Letsencrypt rend pénible la mise en œuvre de certains protocoles de validation du certificat, ou oblige à limiter leur portée, mais ne rend pas impossible leur utilisation. Elle invite surtout les administrateurs de serveur à industrialiser leurs processus.

5. CLIENTS ALTERNATIFS

L'équipe derrière Letsencrypt a fourni le client écrit en python que nous avons pu utiliser plus haut, mais ce dernier n'est qu'un client du protocole ACME que le **service** de Letsencrypt fournit. Il est donc possible de développer d'autres clients dans d'autres langages.

De nombreux développeurs ont écrit leur propre client ACME, et l'équipe de letsencrypt les a énumérés sur son site de support : **https://community.letsencrypt.org/t/list-of-client-implementations/2103/21**.

L'avantage de disposer de librairies mettant en œuvre le protocole ACME est multiple : il permet à chacun de voir comment fonctionne ce protocole, et donc d'en vérifier la sécurité. Il permet de limiter les risques systémiques : souvenez-vous des récentes failles (nombreuses) d'OpenSSL qui ont mis l'industrie de la sécurité sens dessus dessous...

De plus, les librairies clientes du protocole ACME permettent de disposer du service fourni par Letsencrypt dans des contextes industriels : les professionnels de l'Internet peuvent désormais ajouter dans leurs produits le chiffrement TLS accompagné de certificats valides. Aussi, les fournisseurs de service de mail, de chat, d'hébergement web, de logiciel de gestion de serveurs, n'ont plus de raison de ne pas mettre en œuvre TLS dans les services qu'ils fournissent.

6. ...ET DEMAIN ?

L'arrivée de Letsencrypt est donc un événement majeur dans le monde de l'Internet et du Web. De par son aspect totalement gratuit et automatisable, il rend possible l'industrialisation de la sécurité des communications sur Internet. D'ici quelques années, le temps que le monde du logiciel s'adapte et intègre ce protocole, on peut donc espérer que la généralisation de TLS soit une réalité, et que des logiciels clients comme Firefox ou Chrome décident d'utiliser HTTPS par défaut avant HTTP, sécurisant un peu mieux le Web. Ils ont d'ailleurs déjà commencé à restreindre certaines fonctions (comme la géolocalisation) à HTTPS **[9]**. Il devrait en être de même pour les autres protocoles comme SMTP, IMAP ou XMPP.

Cependant, en dehors de cet optimisme béat, rappelons que la sécurité informatique est souvent le parent pauvre de l'Internet, ne bénéficiant de progrès que lorsque les risques sont très élevés. L'intégration de Letsencrypt, en dehors de logiciels ou d'hébergeurs majeurs, risque donc de prendre longtemps. Si vous avez moyen d'améliorer la sécurité et la confidentialité des correspondances dans votre emploi actuel, idéalement chez un acteur important de l'industrie du Web ou de l'Internet, foncez ! ■

RÉFÉRENCES

[1] **https://cabforum.org/**

[2] **https://www.eff.org/https-everywhere**

[3] **https://github.com/letsencrypt/letsencrypt/graphs/contributors**

[4] Ce script est disponible en ligne à l'adresse :
https://github.com/octopuce/octopuce-goodies/tree/master/letsencrypt-renew

[5] **https://fr.wikipedia.org/wiki/Agrafage_OCSP**

[6] **https://en.wikipedia.org/wiki/DigiNotar**

[7] **https://googleonlinesecurity.blogspot.fr/2015/10/sustaining-digital-certificate-security.html**

[8] **https://community.letsencrypt.org/t/please-avoid-3-0-1-and-3-0-2-dane-tlsa-records-with-le-certificates/7022**

[9] **https://codereview.chromium.org/1530403002/**

DÉTECTER L'INTERCEPTION DES FLUX WEB CHIFFRÉS

Rexy, Raphaël PION & Hugo MEZIANI

Aujourd'hui, nous accordons une confiance quasi aveugle au protocole HTTPS de chiffrement de flux web. À l'heure du « tout connecté », il est important de mesurer les risques auxquels on s'expose en se connectant au moyen de ce protocole. Les menaces d'interception sont présentes et identifiées. Comment détecter qu'elles sont actives ?

Dans cet article, nous souhaitons que le lecteur s'interroge sur la sécurité effective du protocole de transport des flux HTTPS implémenté dans tous les clients web. Si l'encapsulation HTTP avec le protocole SSL/TLS utilise un matériel cryptographique sûr et fiable, qu'en est-il de l'application que l'on en fait sur Internet ? Quelles sont les limites de la confiance à accorder aux flux web chiffrés ? Pour y répondre, nous commencerons par rappeler le fonctionnement de TLS (*Transport Layer Security),* anciennement SSL (*Secure Socket Layer*). Nous soulèverons l'ambiguïté suivante : nous authentifions un serveur web grâce au certificat numérique qu'il nous présente. Pourquoi le serveur web ne fait-il pas la même chose de son côté ? Quels problèmes cela pose-t-il ?

Aussi, nous mettrons en évidence trois différents types d'interception du trafic SSL/TLS. Le premier type est historique. Il exploite le principe de pollution du cache ARP des équipements réseau puis du dechiffrement du trafic au moyen de faux certificats (Man in The Middle). Un deuxième type d'interception est lié aux *appliances* de sécurité déployées aux frontières des réseaux d'entreprise. Ces équipements dédiés à la sécurité sont souvent configurés (volontairement ou non) pour réaliser de l'inspection SSL/TLS. Cela peut poser quelques problèmes dans l'entreprise quand cette inspection (qui est une véritable interception) est réalisée à l'insu des utilisateurs et parfois même du Responsable de la Sécurité des Systèmes d'Information (RSSI). Un troisième type d'interception est apparu avec les versions récentes des logiciels anti-malwares. Dans un but qui « peut sembler » légitime, des logiciels comme **Avast [1]** ou encore **Kaspersky** pratiquent l'interception HTTPS pour protéger l'utilisateur d'un vecteur d'intrusion de logiciel malveillant.

À la fin de cette partie, nous verrons que dans un but légitime, l'inspection de flux HTTPS permet de sécuriser ce canal chiffré qui est peut être vecteur d'infection ou encore de s'assurer que les utilisateurs d'un réseau ne naviguent pas sur des sites blacklistés.

Pour lutter contre l'interception de flux web chiffrés non légitimes, les clients web ont mis en place quelques protections que nous détaillerons.

Enfin, nous présenterons une méthode permettant à tout internaute de détecter une interception SSL/TLS. Nous avons développé une implémentation de cette méthode sous forme d'une extension au navigateur web Firefox que vous pouvez tester dès à présent.

1. FLUX WEB CHIFFRÉ

Le SSL ou TLS sont des protocoles de sécurisation permettant d'échanger des informations entre deux équipements. Ils sont situés entre les couches transport et application du modèle OSI (couche 4 à 7). Ces protocoles ajoutent une ou plusieurs fonctions de sécurité (confidentialité, intégrité, authenticité, imputabilité). Le protocole SSL a été développé par Netscape. Par la suite, le développement a été repris par l'*Internet Engineering Task Force* (IETF) qui a rebaptisé ce protocole : « TLS ».

Ce protocole est utilisé entre un client et un serveur. Plusieurs étapes sont nécessaires à sa mise en place effective :

- Le client (souvent un navigateur) demande au serveur la mise en place d'une connexion sécurisée par SSL/TLS. Il envoie la version SSL/TLS utilisée (la dernière version stable étant TLS 1.2) et les algorithmes de chiffrement qu'il peut exploiter (ex : RC4, Triple DES, AES, IDEA, DES ou Camelia). Cet ensemble constitué de la version du protocole et des algorithmes exploitables est appelé le « matériel cryptographique ».

- Le serveur envoie un certificat de sécurité contenant sa clé publique ainsi que des informations relatives à son identité et ses capacités cryptographiques. Le serveur peut aussi demander le certificat de sécurité du client si l'authentification de ce dernier est requise (ce qui n'est jamais demandé lors de connexions sur des serveurs publics).
- Si le serveur ne peut pas s'authentifier correctement (autorité de certification non reconnue par le navigateur du client), l'utilisateur est prévenu et la connexion chiffrée peut être avortée (alerte de certificat non signé).
- Si le client n'a pas besoin de s'authentifier, il va créer le *pre-master secret* selon le matériel cryptographique utilisé. Celui-ci sera chiffré de manière symétrique avec la clé publique du serveur et sera envoyé au serveur. Si une demande d'authentification a été demandée, le client envoie son certificat, signe un message avec sa clé privée et l'envoie au serveur. La clé symétrique sera chiffrée avec la clé publique du serveur.
- Si le client ne doit pas s'authentifier, le serveur va déchiffrer le *pre-master secret* avec sa clé privée (fichier au format « .key ») afin de générer avec le client, le *master secret*. Si le client doit s'authentifier, le déchiffrement se fera avec la clé publique du client et le serveur effectuera l'étape précédente faite par le client.
- La clé de session sera générée avec le *master secret* pour le client et le serveur. Ces deux clés sont identiques et serviront au chiffrement et au déchiffrement des communications. Le *handshake* est terminé.
- La connexion TLS est maintenant établie, les échanges entre le client et le serveur sont chiffrés.
- À la fin de la connexion, la clé de session sera révoquée par le serveur.

No.	Time	Source	Destination	Protocol	Length	Info
98	2.235867000	10.105.200.167	198.41.208.124	TCP	66	34622→443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
111	2.256459000	198.41.208.124	10.105.200.167	TCP	66	443→34622 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=1024
112	2.256554000	10.105.200.167	198.41.208.124	TCP	54	34622→443 [ACK] Seq=1 Ack=1 Win=16384 Len=0
121	2.261727000	10.105.200.167	198.41.208.124	TLSv1.2	571	Client Hello
140	2.285198000	198.41.208.124	10.105.200.167	TCP	56	443→34622 [ACK] Seq=1 Ack=518 Win=30720 Len=0
142	2.285200000	198.41.208.124	10.105.200.167	TLSv1.2	206	Server Hello, Change Cipher Spec, Encrypted Handshake Message
146	2.286401000	10.105.200.167	198.41.208.124	TLSv1.2	105	Change Cipher Spec, Hello Request, Hello Request
158	2.291064000	10.105.200.167	198.41.208.124	TLSv1.2	111	Application Data

Fig. 1 : Capture avec Wireshark d'un handshake TLS.

Au même titre qu'un notaire assure l'authenticité d'actes administratifs, il existe des Autorités de Certification (AC) pour les transactions numériques. Les navigateurs ont connaissance de ces AC à travers leurs certificats stockés dans des magasins qu'ils gèrent (cas de **Firefox**) ou gérés par le système d'exploitation (cas d'**Internet Explorer** sous Windows).

Quand ces navigateurs se connectent à un serveur web sécurisé, ils reçoivent le certificat d'identité du serveur. Ils vérifient que ce dernier est bien signé par une autorité présente dans leur magasin.

Deux cas d'usage peuvent apparaître :

- Le premier cas : le certificat présenté est authentifié.

Fig. 2 : Certificat accepté par le navigateur Firefox.

Ce cadenas indique que le certificat du site consulté est signé (donc reconnu) par une autorité de la liste des AC. Le navigateur initie la connexion chiffrée avec le serveur. Ce n'est pas pour autant que l'on peut affirmer que la connexion est sécurisée. En effet, quelle confiance pouvons-nous donner aux AC présentes dans le magasin des certificats. Dans le cas d'un poste de travail en entreprise, des certificats d'autorité peuvent être intégrés sur les machines clientes de manière automatique via les outils de gestion de parc.

⇨ Le deuxième cas : le certificat présenté n'est pas signé par une autorité reconnue.

Fig. 3 : Alerte « certificat autosigné » émise par le navigateur Firefox.

Dans ce cas, le navigateur demande à l'utilisateur de confirmer (en comprenant les risques) avant d'initier la connexion chiffrée. Le risque est qu'une personne ayant intercepté le flux ait présenté son propre certificat (autosigné ou *pushé* sur la machine victime). Elle est alors en capacité de déchiffrer le contenu de la communication. On parle d'une attaque de type « homme du milieu » (*Man In The Middle* ou MITM).

Une des fonctionnalités de TLS est de pouvoir authentifier le client à partir d'un certificat qu'il doit présenter au serveur. Malheureusement cette méthode est complexe à mettre en place pour le grand public. La non-authentification du client par le serveur donne donc la possibilité d'une interception SSL expliquée ci-après.

2. INTERCEPTIONS SSL/TLS

Nous allons vous présenter dans cette partie les différents cas de figure où nous pouvons être soumis à une interception SSL/TLS.

2.1 En local, dans nos logiciels

Dans un but vraisemblablement bienveillant, certains antivirus déchiffrent les connexions SSL/TLS de leurs systèmes hôtes afin d'anticiper le téléchargement de maliciels. Certains d'entre eux le font de manière native (sans demande explicite à l'utilisateur...). Ces logiciels ont donc accès en clair à tous les flux chiffrés de l'utilisateur. Quand on sait que parallèlement, ils sont connectés au site des éditeurs pour récupérer les mises à jour... En fait, cela ne devrait pas nous inquiéter, car ils ont de toute façon tous les pouvoirs sur nos machines.

Voici, en première approche l'état de présence de cette caractéristique pour quelques antivirus :

Logiciel Antivirus	SSL Inspection	Par défaut
Avast !	OUI	OUI
Kaspersky	OUI	OUI (sur certains sites)
BitDefender	OUI	NON
ESET	OUI	NON

Pour effectuer cette vérification, ces antivirus installent un certificat racine (au-dessus des autres autorités de certification) dans les magasins de certificats. Ensuite, via un proxy web local, ils présentent au navigateur un faux certificat du serveur web consulté généré à la volée.

Si cette pratique peut s'avérer utile en termes d'analyse virale de flux chiffrés, il existe cependant de mauvaises applications. Dans un premier temps, prenons l'exemple du fameux *adware* **Superfish [2]**, installé par défaut sur certains **Lenovo**. Ce programme déchiffre le flux HTTPS pour en analyser le contenu et ainsi proposer ses propres publicités. Notons que **Superfish** utilise le même certificat et la même clef sur tous les clients.

Plus récemment l'*adware* **PrivDog [3]** qui, dans l'ensemble a les mêmes fonctionnalités que **Superfish**, crée un certificat et une clef sur toutes ces installations. Cet *adware* intercepte tous les

certificats serveur et les remplace par le sien (signé par son propre certificat racine). Cela veut donc dire que PrivDog va faire accepter au navigateur des connexions SSL/TLS avec des sites ne possédant pas de certificats serveur valides. Le client web acceptera maintenant tous les certificats serveur qu'ils soient signés par une autorité reconnue ou pas.

Penchons-nous maintenant sur les produits de sécurité précédemment cités. En interceptant le trafic HTTPS, l'antivirus ne fait rien d'autre que de créer une connexion TLS et vérifier le certificat d'un site web. En d'autres termes, il se comporte comme un navigateur web. Encore faut-il correctement implémenter ces protocoles. En effet, la majorité des logiciels appliquant une inspection SSL/TLS empêche la protection *HTTP Public Key Pinning (HPKP)* de fonctionner. Cette fonctionnalité permet de référencer dans les navigateurs les clefs publiques des serveurs web. Nous détaillerons cette protection dans une partie suivante. Mais les navigateurs ont choisi de faire une concession dans l'intégration de cette protection. Elle ne sera pas effective sur les certificats racines installés manuellement. Comprenez donc, les certificats installés par les différents logiciels nommés plus haut. Nous pouvons aussi être confrontés à une mauvaise implémentation de TLS. En effet, l'antivirus **Kaspersky** était vulnérable à l'attaque *Factoring RSA Export Keys (FREAK)* **[4]** qui permet à un attaquant en MITM de factoriser une clef de chiffrement RSA d'une longueur de 512 bits ou moins. **Kaspersky** encore, rend le navigateur du client vulnérable à un vol de session en activant la compression de paramètre TLS avec l'algorithme Deflate. Cette attaque se nomme : *Compression Ratio Info-leak Made Easy (CRIME)* **[5]**. Durant nos tests, cette vulnérabilité potentiellement exploitable était toujours présente dans la version **Kaspersky** Anti-Virus 16.0.0.614.

Attention !

Nous invitons les lecteurs à tester leurs clients web sur des sites dédiés comme https://howsmyssl.com afin de déceler une mauvaise implémentation de SSL/TLS dans les modules d'interception.

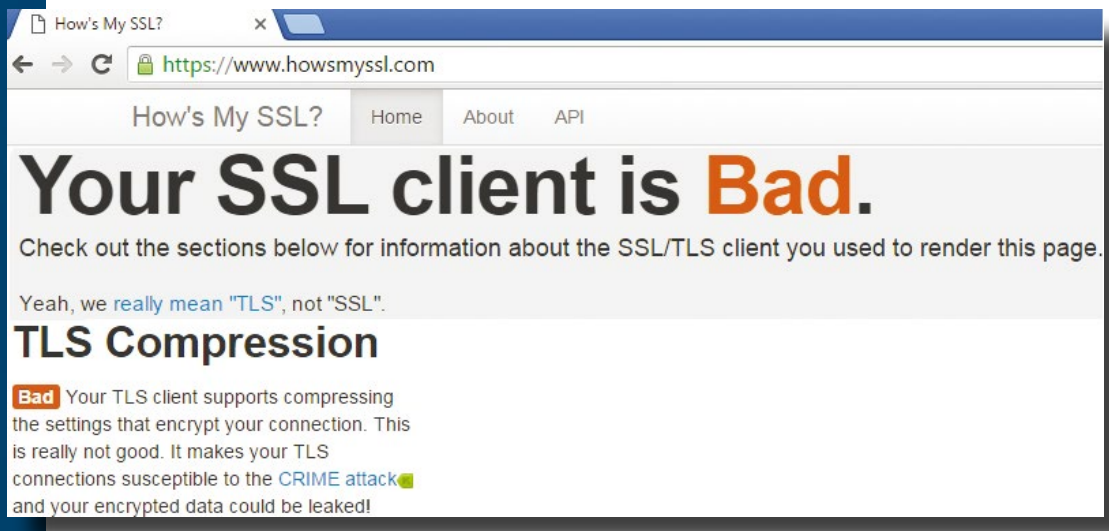


Fig. 4 : Module d'inspection SSL de Kaspersky présentant la vulnérabilité « CRIME ».

De nombreux logiciels désirent déchiffrer nos connexions à des fins bienveillantes en assurant un service de protection complet. Force est de constater que leurs implémentations abaissent le niveau de sécurité globale du protocole HTTPS **[6]**.

2.2 Sur un réseau privé

Dans une attaque de type « MITM », le but de l'attaquant est de récupérer le trafic de sa victime. Pour cela, il peut exploiter deux techniques de détournement de flux (cf. Figure 5). Une fois le trafic récupéré, il peut directement lire les protocoles non chiffrés (FTP, HTTP, VOIP, etc.). Pour les flux chiffrés, l'attaquant doit fabriquer un « faux » certificat pour le présenter à sa victime qui constatera une alerte dès qu'elle se dirigera vers un site en HTTPS. Comme les certificats comportent des champs spécifiques à chaque serveur (nom de domaine, date, etc.), l'attaquant doit être capable de les générer dynamiquement au rythme de la navigation de sa victime. Ces certificats sont autosignés par une AC qui a été créée pour le besoin de l'attaque. Des outils comme **sslsplit [7]** permettent d'automatiser cette tâche.

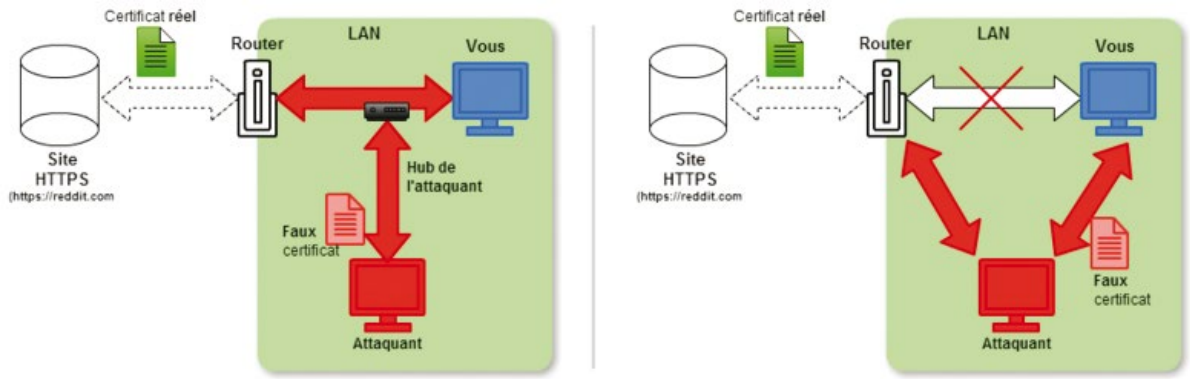


Fig. 5 : (à droite) Détournement de flux en modifiant les caches ARP. (à gauche) Détournement de flux en intercalant un HUB sur le réseau.*

Il est de plus en plus difficile de mener ce genre d'attaques grâce aux protections telles que HSTS que nous détaillerons plus tard.

2.3 Sur les réseaux d'entreprises

En entreprise, cette pratique peut être mise en place via les proxys intégrés aux appliances de sécurité déployés en interface des accès Internet (figure 6, page suivante). En France, cette surveillance de connexion chiffrée doit être explicitée dans la charte informatique signée par l'employé. L'appliance de sécurité est intercalée entre le réseau interne et l'accès Internet afin de filtrer les connexions entrantes et sortantes. Son but premier est d'appliquer des règles pour protéger le réseau de l'entreprise et éviter des fuites d'information. Parfois, le déchiffrement des connexions TLS est effectué sans que le client en soit informé. Parfois, l'entreprise ne sait même pas que son appliance possède cette fonctionnalité et qu'elle est activée.

Contrairement à un « MITM » via **sslsplit** avec certificats autosignés générés à la volée, ici l'administrateur n'aura aucun mal à distribuer à toute la flotte d'ordinateurs du réseau le certificat de l'appliance en utilisant Active Directory et un Group Policy Object. La GPO permet de déployer les configurations liées à la sécurité, notamment des certificats serveur.

Note

***La technique de détournement de flux dite de « cache poisonning » peut être évitée en configurant correctement certains équipements réseau récents (inspection ARP sur les commutateurs, « isolation client » pour les points d'accès WIFI, etc.) ou bien les systèmes d'exploitation en bloquant les réponses « gratuitous ARP ».**

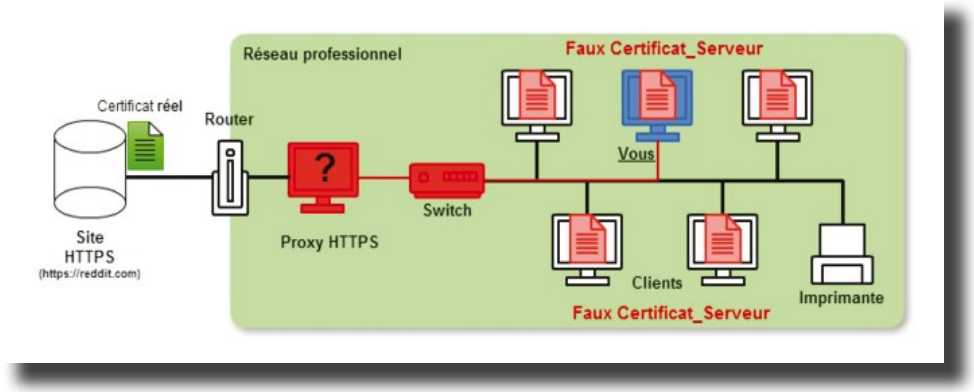


Fig. 6 : Exemple d'un réseau d'entreprise où l'inspection SSL est présente.

Il existe un document écrit par l'ANSSI expliquant comment organiser une inspection SSL/TLS sur un réseau d'entreprise **[8]**.

3. AMÉLIORATIONS SSL/TLS

Pour se protéger de ce type de pratiques quand elles ne sont pas légitimes, des protections existent. Nous allons vous en présenter quelques-unes implémentées soit du côté client, soit du côté serveur.

3.1 HTTP Strict Transport Security (HSTS)

HTTP Strict Transport Security est un mécanisme de sécurité à intégrer au serveur web. Il permet d'indiquer aux clients (navigateur web) compatibles de ne communiquer avec lui qu'en utilisant une connexion chiffrée comme HTTPS. HSTS est régie par le temps. La période durant laquelle un client ne devra accéder à un serveur uniquement via HTTPS est indiquée dans l'en-tête nommé *Strict-Transport-Security : max-age=xxx*.

La figure 7 nous montre la négociation entre un client et un site avec et sans HSTS.

Quand HSTS est activé sur un serveur, l'agent web procède de la façon suivante :

- ⇨ Il remplace tous les liens **http://exemple.com** par **https://exemple.com**.
- ⇨ Face à un certificat autosigné (non reconnu), il affiche un message d'erreur et interdit l'accès au serveur.

Ce protocole de sécurité est très efficace contre des attaques de type MITM avec génération de certificats autosignés ou encore contre des écoutes passives, car la connexion sera toujours redirigée en HTTPS.

HSTS présente néanmoins un défaut. Il ne protège un client qu'après sa première connexion. En effet si un équipement est déjà intercalé sur le réseau de son client durant sa première connexion à un site web, le client enregistrera le certificat de l'équipement comme « le vrai certificat du site web » si celui-ci est enregistré sur la machine du client comme certificat de confiance. Pour pallier ce problème, il existe un mécanisme de préchargement : les sites proposant HSTS peuvent s'enregistrer **[9]** sur une liste incluse dans les navigateurs afin d'assurer l'authenticité des paramètres de HSTS.

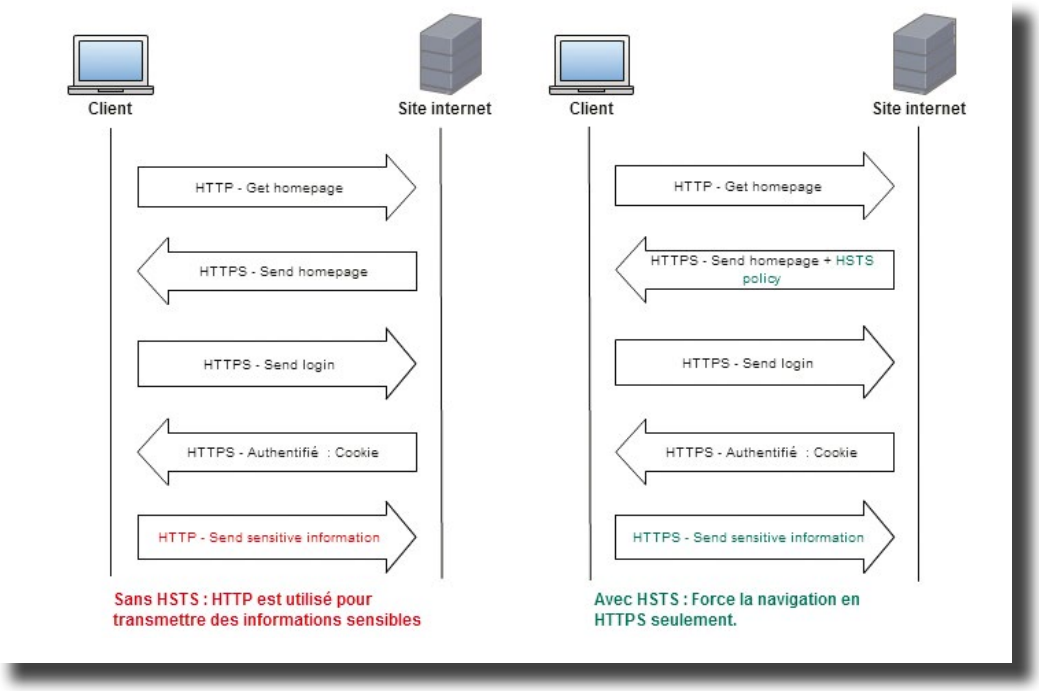


Fig. 7 : Échange entre clients serveur avec et sans HSTS.

Aujourd'hui, Google Chrome, Chromium, Firefox, Opera et Internet Explorer supportent le HSTS.

3.2 HTTP Public Key Pinning (HPKP)

HTTP Public Key Pinning est un moyen de protection des sites internet contre l'usurpation de certificats autosignés ou bien émis par des autorités de certification compromises. Ce protocole est défini par la RFC 7469.

Durant la première connexion, le site va présenter une liste contenant les clés publiques de confiance. Cette liste est ensuite mémorisée par le navigateur. Par la suite, le client web refusera la connexion si la clé publique diffère de celles mémorisées par le navigateur.

HPKP est supporté par **Google Chrome** et **Firefox**, mais pas par **Internet Explorer**/**Edge**.

3.3 Matériel cryptographique

Avec l'avancée de la puissance de calcul de nos processeurs, nous devons adapter la taille des clefs cryptographiques. Si la fiabilité mathématique des algorithmes de chiffrement est acquise, nous pouvons faire face à des utilisations de clefs de taille insuffisante.

Vous pouvez auditer votre propre serveur afin de connaître l'état de son matériel cryptographique. La commande suivante indique le matériel cryptographique proposé par le module SSL/TLS de votre serveur web. Durant nos expérimentations, nous sommes tombés sur des sites sensibles (banques, opérateurs) présentant un matériel cryptographique utilisant des algorithmes faibles (voire très faibles).

Terminal

```
$ nmap --script ssl-enum-ciphers -p 443 exemple.com
[...]
|   TLSv1.2:
|     ciphers:
|       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA - strong
|       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 - strong
|       TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 - strong
|       TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA - strong
|       TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 - strong
|       TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 - strong
|       TLS_RSA_WITH_3DES_EDE_CBC_SHA - strong
|       TLS_RSA_WITH_AES_128_CBC_SHA - strong
|       TLS_RSA_WITH_AES_256_CBC_SHA - strong
|       TLS_RSA_WITH_DES_CBC_SHA - weak
|       TLS_RSA_WITH_IDEA_CBC_SHA - weak
|       TLS_RSA_WITH_RC4_128_MD5 - strong
|       TLS_RSA_WITH_RC4_128_SHA - strong
|     compressors:
|       NULL
|_  least strength: weak
```

Vous pouvez aussi constater le magasin cryptographique de votre navigateur favori grâce à des sites comme : **https://cc.dcsec.uni-hannover.de**.

4. DÉTECTION D'UNE INTERCEPTION SSL/TLS

Nous pouvons détecter une interception en comparant le certificat reçu par le client web avec celui que recevrait un autre client web situé à l'extérieur du réseau où l'interception a lieu (un équipement de confiance externe). Dans le cas normal, il n'y a qu'un seul certificat et son contenu est le même côté client web et côté équipement de confiance. Mais si l'empreinte du certificat diffère, cela veut dire que la communication a sûrement été déchiffrée entre le client web et le serveur consulté !

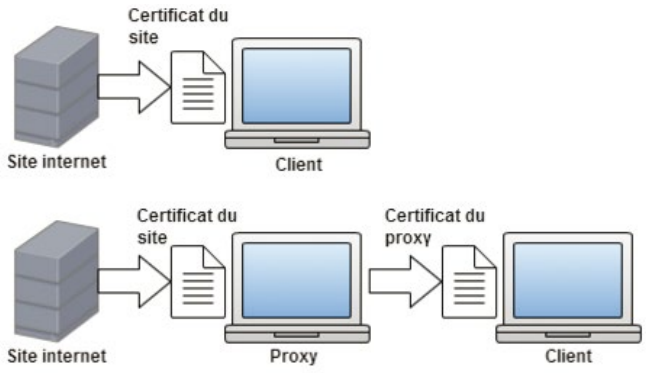


Fig. 8 : Présentation du certificat avec ou sans proxy.

Dans la figure 8, nous voyons que le certificat reçu par le client est celui du proxy. Dans un premier temps, ce proxy exécute la requête du client. Dans un second temps, il devient le client du site internet, et retransmet les réponses du site web au client en se faisant passer pour le serveur web.

4.1 CheckMyHTTPS

Dans le cadre d'un projet piloté par le laboratoire de Cryptologie et Virologie Opérationnelles de l'**ESIEA**, nous avons développé un démonstrateur qui permet aux internautes de mettre en évidence une interception HTTPS. Ce démonstrateur a été développé sous la forme d'une extension *open source* au navigateur web **Firefox.**

CheckMyHTTPS a été signé par la fondation Mozilla. Il fonctionne de la manière suivante dans sa dernière version (V2.0.4) :

- À l'ouverture du navigateur, l'extension effectue une première vérification HTTPS directement sur notre serveur « **https://checkmyhttps.net** ».
- L'utilisateur peut à tout moment *checker* sa connexion HTTPS sur un site en particulier en un clic sur l'icône de l'extension. L'empreinte SHA256 du certificat du serveur est alors extraite et envoyée sur notre serveur.
- Ce serveur demande alors lui aussi le certificat du site HTTPS consulté par l'utilisateur et il en calcule aussi l'empreinte numérique. Une comparaison est alors réalisée entre les deux empreintes et le résultat est envoyé à l'extension du navigateur de l'utilisateur.

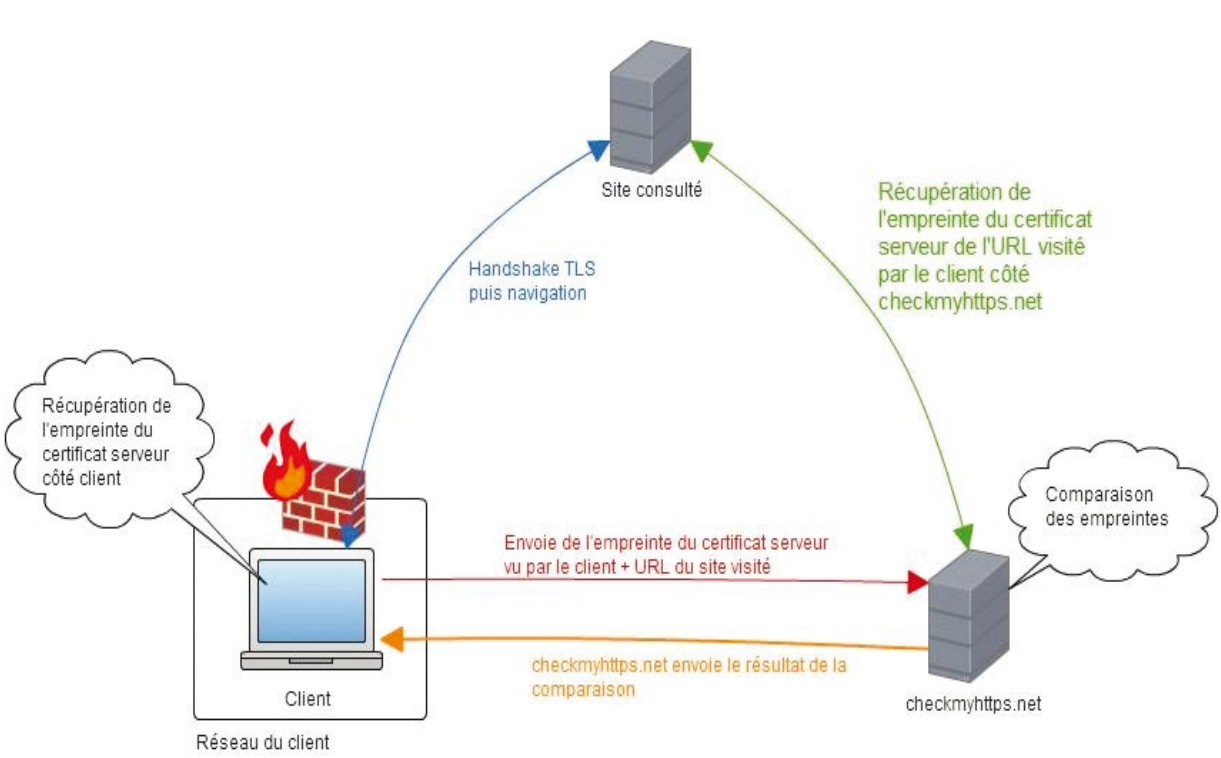


Fig. 9 : Fonctionnement de CheckMyHTTPS.

- Afin de contourner le problème concernant les sites HTTPS présentant plusieurs certificats serveur (**comme Google ou YouTube**), nous effectuons un deuxième test sur des serveurs de confiance (réputés de ne présenter qu'un seul certificat).
- L'extension « **CheckMyHTTPS** » du navigateur affichera la réponse sous la forme d'un cadenas : vert si la connexion est sécurisée ; rouge si la connexion semble surveillée.

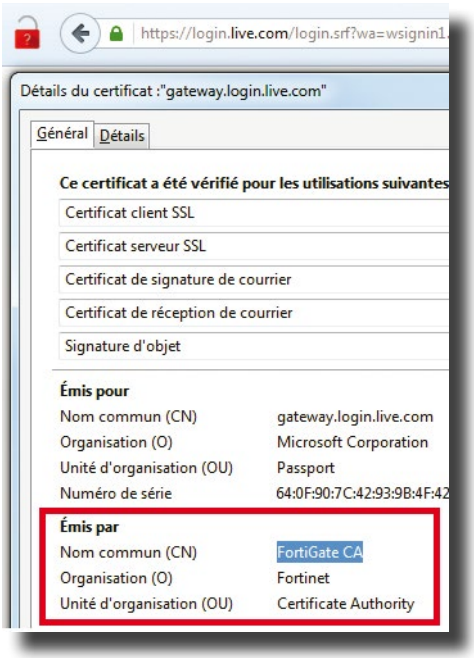


Fig. 10 : Client intercepté par Fortigate240D.

Tout internaute peut savoir si le flux SSL entre son navigateur et un site web sécurisé est intercepté, voire modifié. À titre d'exemple, ce démonstrateur réagit à l'inspection SSL de Fortigate 240D.

Sur la figure 10, vous constaterez une inspection SSL mise en évidence par l'extension **CheckMyHTTPS** en haut à gauche (cadenas rouge). Vous remarquerez le cadenas vert à côté de l'URL indiquant que le certificat de Fortigate est reconnu par le navigateur. Celui-ci a été manuellement installé sur la machine cliente.

Sur le réseau d'une entreprise, l'employé pourra demander au Responsable des Systèmes d'Information (RSI ou DSI) les raisons qui l'ont poussé à installer un tel dispositif. Il pourra vérifier la conformité de la charte Internet. Il sera en tout cas conscient des risques quand il surfe de manière « sécurisée »... **[10]**.

CONCLUSION

Il y a quelques années, un scandale a été mis au jour. Google a publié une note informant qu'un certificat signé par l'ANSSI avait été utilisé pour (selon Google) attaquer ses services. En réalité, l'ANSSI a généré un certificat de sécurité pour inspecter le trafic HTTPS du Ministère des Finances (les utilisateurs en ayant été informé). Les administrateurs de ce ministère voulaient filtrer les flux des webmails tels que Gmail afin d'empêcher d'éventuelles menaces ou fuites de données. La véritable question à se poser est comment Google a-t-il pu détecter cette interception dans un réseau privé ? Nous vous invitons à lire les deux articles suivants pour plus de détails **[11][12]**.

Récemment, **Microsoft** a ajouté secrètement plusieurs certificats racines à sa *Certificate Trust List* (CTL) appartenant à **Amazon**. Ils apparaissent juste au moment où le service de certification d'Amazon est créé. N'oublions pas les liens étroits entre Amazon et les agences de renseignement **[13]**. La protection du trafic Internet mondial est régie par une quarantaine d'entreprises étatiques. Devons-nous leur faire confiance aveuglément ? Il existe néanmoins de nouvelles solutions comme **Let's Encrypt [14]** qui propose un service de certification gratuit et transparent pour serveur web.

Dans cet article, nous voulions montrer aux lecteurs les limites de la sécurité du protocole de chiffrement le plus utilisé sur Internet. Si les algorithmes de cryptographie sont à ce jour considérés comme mathématiquement fiables, leurs implémentations et leurs utilisations sur Internet présentent des failles liées à la non-authentification du client ou encore à la confiance aveugle que nous donnons aux autorités supérieures de certification. ▮

REMERCIEMENTS

Un grand merci à Rexy, Éric Filiol, Paul Irolla et Maëva Dupuy pour leurs relectures attentives.

RÉFÉRENCES

[1] David De Oliveira, « Avast : un espion dans votre PC ? » SecuriteOff, 2015.

[2] Superfish : **https://en.wikipedia.org/wiki/Superfish**

[3] Software Privdog worse than Superfish :
https://blog.hboeck.de/archives/865-Software-Privdog-worse-than-Superfish.html

[4] CVE-2015-0204 :
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-0204

[5] CVE-2012-4929 :
https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2012-4929

[6] Electronic Frontier Foundation :
https://www.eff.org/deeplinks/2015/02/dear-software-vendors-please-stop-trying-intercept-your-customers-encrypted

[7] sslsplit : **https://www.roe.ch/SSLsplit**

[8] ANSSI Note Technique :
http://www.ssi.gouv.fr/uploads/IMG/pdf/NP_TLS_NoteTech.pdf

[9] Préchargement HSTS : **https://hstspreload.appspot.com/**

[10] CheckMyHTTPS : **https://checkmyhttps.net**

[11] MITM de Google par l'ANSSI :
https://reflets.info/mitm-de-google-par-l-anssi-la-theorie-du-doigt-qui-pointait-la-lune/

[12] ANSSI MITM :
http://securityaffairs.co/wordpress/20325/hacking/french-anssi-mitm.html

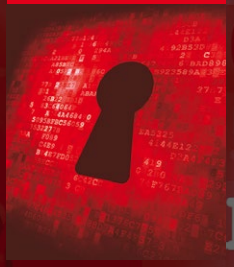
[13] Amazon & CIA :
http://www.salon.com/2014/12/01/amazons_frightening_cia_partnership_capitalism_corporations_and_our_massive_new_surveillance_state/

[14] Let's Encrypt : **https://letsencrypt.org/**

3

3 LES ENJEUX DE LA NORMALISATION

À découvrir dans cette partie...

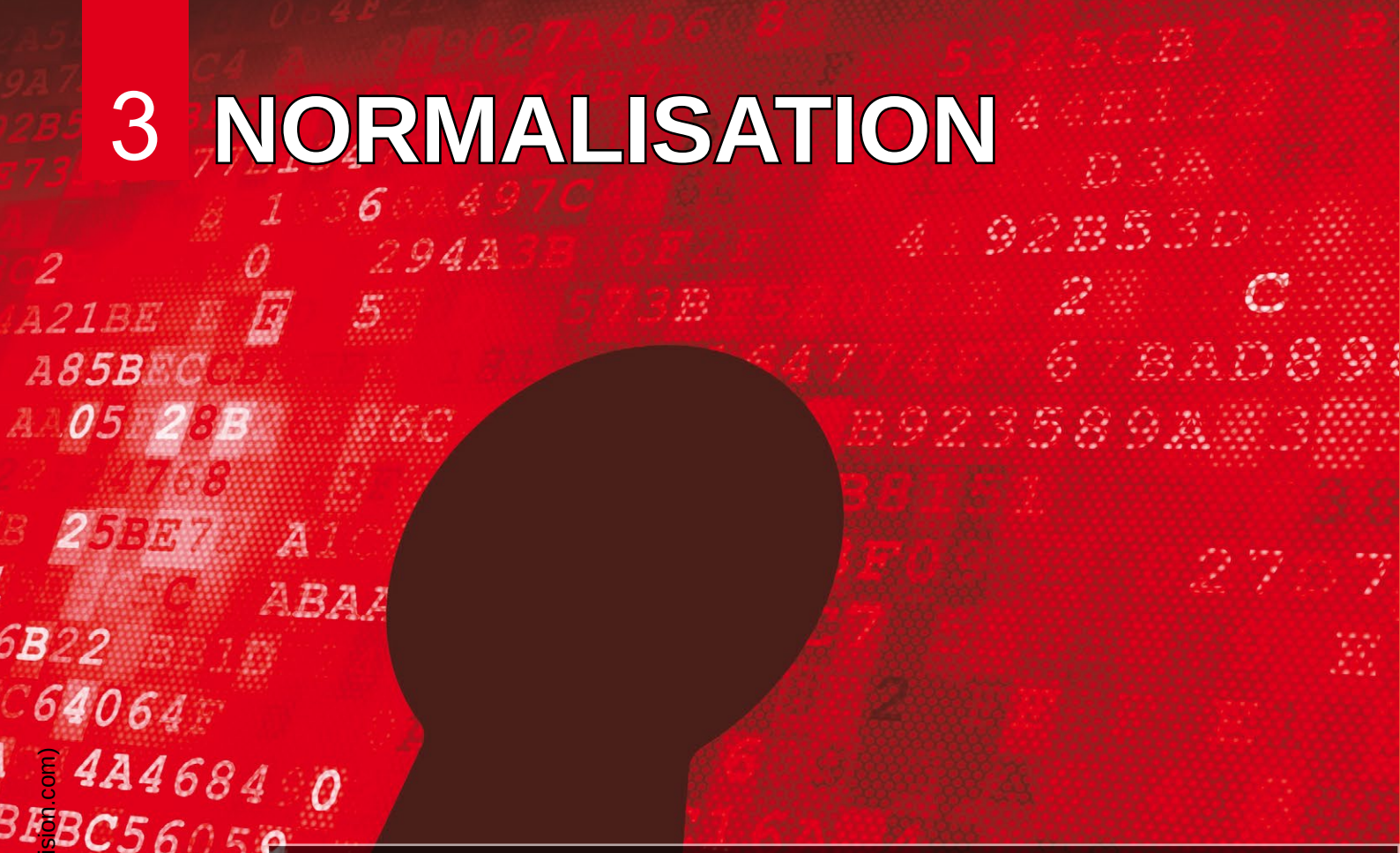


3.1 Standardisation des courbes elliptiques : à qui faire confiance ?

Les courbes qui seront utilisées dans la prochaine version de TLS ont fait intervenir durant leur sélection par l'IETF une composante nouvelle : la confiance dans le processus de génération de la courbe. Découvrez ce nouvel aspect. p. 84

3.2 And you make crypto, and I make crypto, everybody's making crypto !

Vous souhaitez utiliser de la cryptographie directement dans un navigateur avec JavaScript par le biais d'une API normalisée ? Voici les derniers travaux du W3C en la matière qui visent à développer un tel outil. p. 94



STANDARDISATION DES COURBES ELLIPTIQUES : À QUI FAIRE CONFIANCE ?

gapz

La cryptographie à base de courbes elliptiques est devenue ces dernières années une des alternatives les plus intéressantes à RSA en termes de cryptographie asymétrique. Cependant, sa jeunesse et ses développements récents en font encore une solution questionnée, tant sur sa solidité que sur la manière de correctement l'utiliser. L'heure est aujourd'hui à une nouvelle vague de normalisation dans une situation pour le peu troublée par un élément clef du succès d'un standard: la confiance.

INTRODUCTION

La cryptographie à base de courbes elliptiques a été introduite il y a maintenant plus de trente ans et l'intérêt de son usage ne fait aujourd'hui plus de doute. D'une part, les algorithmes de décomposition en produits de facteurs premiers ont été largement améliorés ces dernières années, rendant ainsi certaines tailles de clefs RSA obsolètes et obligeant par là même à augmenter sensiblement la taille des clefs. D'autre part, le meilleur algorithme connu pour calculer le logarithme discret dans un groupe multiplicatif (une structure mathématique classique très utilisée en cryptographie) ne s'applique pas pour la plupart des courbes elliptiques. Du fait de sa solidité, les tailles de clefs nécessaires sont plus petites et rendent également les coûts de calculs plus faibles que d'autres systèmes tels que RSA. D'abord largement breveté et standardisé par des organismes états-uniens, le domaine des courbes elliptiques a connu dernièrement des développements importants, notamment avec l'introduction de nouvelles courbes (Edwards), le développement d'attaques contre les implémentations et l'usage désormais régulier de celles-ci avec par exemple ECDHE ou ECDSA dans TLS. Seulement, il apparaît aujourd'hui primordial de développer des standards reconnus pour leur transparence, ce qui n'est entre autres pas ou plus le cas des standards du NIST. En effet, au-delà même du fait que des mathématiciens de la NSA soient à l'origine des courbes du NIST, la compromission du standard Dual_EC_DRBG (du NIST également) montre bien qu'il est impératif d'être en pleine maîtrise des choix relatifs à la structure d'un algorithme en cryptographie (et de ses constantes bien sûr).

C'est ainsi que ces derniers mois ont vu naître beaucoup de recherches et de discussions autour des courbes à normaliser. L'enjeu étant de taille, cela va certainement définir quelles courbes vont se retrouver massivement utilisées et remplacer peu à peu les vieilles courbes du NIST (même si ces dernières sont encore considérées comme sûres). La sécurité des courbes repose sur plusieurs critères mathématiques qui sont à l'heure actuelle majoritairement partagés par la communauté de la cryptographie. La tension principale autour de la sélection des courbes à normaliser ne se joue pas sur ce dernier terrain, mais bien sûr celui de l'évaluation des avantages et inconvénients de chaque courbe (performance, résistance aux attaques par canaux auxiliaires, simplicité d'implémentation) et, fait assez nouveau, la transparence (reproductibilité ou vérifiabilité) du processus qui a mené à développer et choisir certains paramètres de courbes plutôt que d'autres. Ce dernier point, que l'on retrouve parfois sous le terme de « rigidité », sera l'objet principal traité dans cet article. Après un bref rappel sur les courbes elliptiques et leur histoire, seront discutées les différentes méthodes existantes pour la procédure de génération des courbes, pour terminer sur différents travaux de normalisation en cours. Cet article cherche volontairement à simplifier les descriptions des structures liées aux courbes elliptiques afin de rendre l'ensemble des explications plus accessible aux non-initiés. Ainsi, que les esprits rigoureux pardonnent par avance les abus de langage et les raccourcis un peu simplistes.

1. RAPPEL SUR L'UTILISATION DES COURBES ELLIPTIQUES

1.1 Notions et logarithme discret

Introduisons tout d'abord quelques rudiments de mathématiques. Ce sujet faisant déjà l'objet d'un article dans le présent hors-série, seront utilisées quelques notions définies par ce dernier en essayant malgré tout d'épurer un maximum les équations. Dans tout cet article, les notations suivantes seront utilisées :

- $y^2 = x^3 + ax + b$ comme équation d'une courbe elliptique classique (forme simplifiée de Weierstrass) ;
- a et b comme les constantes de cette équation ;
- p un nombre premier définissant le corps premier (toutes les opérations se font modulo p) ;

⇨ G un point générateur (point de base) ;

⇨ X,Y des points quelconques de la courbe, dont les coefficients sont définis sur F_p ;

⇨ k un scalaire.

Comme évoqué en introduction, l'intérêt principal des courbes elliptiques est la difficulté de calcul du logarithme discret dans une courbe : il est facile de calculer Y = k * X pour k et X connus, mais il est par contre très difficile de trouver k pour X et Y connus : c'est le calcul du logarithme discret. L'opération notée ici * correspond à la multiplication d'un point de la courbe par un scalaire et produit un autre point sur la courbe. La méthode la plus efficace pour faire ce dernier calcul dans un cadre général (il existe des optimisations dans des cas particuliers) se nomme « pollard-rho » et a été introduite il y a maintenant plus de vingt ans.

1.2 Exemple : échange d'un secret partagé : Diffie-Hellman

L'un des usages les plus courants des courbes elliptiques est la génération d'un secret partagé en utilisant le mécanisme de Diffie-Hellman dans le groupe d'une courbe elliptique. De la même manière que dans le groupe multiplicatif d'un corps fini (le cas « classique »), le protocole se définit comme suit :

⇨ Alice génère un entier aléatoire a puis envoie G * a ;

⇨ Bob génère un entier aléatoire b puis envoie G * b ;

⇨ Alice et Bob utilisent le nouveau secret ainsi généré (G * a) * b (égal à (G * b) * a).

L'ensemble des opérations se faisant dans le groupe de la courbe utilisée, c'est-à-dire en fonction des paramètres de celles-ci : un point de base, un grand nombre premier et des constantes liées à la structure de la courbe. Par exemple, si l'on décide d'utiliser la courbe Curve25519 **[1]** alors les paramètres sont les suivants :

⇨ équation de la courbe : $y^2 = x^3 + Ax^2 + x$ (forme de Montgomery) ;

⇨ nombre premier : $2^{255} - 19$;

⇨ point de base : x = 9 ;

⇨ constantes A = 486662.

On retrouve ce mécanisme sous le nom X25519 (pour l'utilisation de la courbe Curve25519 et du mécanisme Diffie-Hellman). Cette primitive est notamment utilisée dans des applications telles qu'OpenSSH, Tor ou Signal. La question centrale qui va être traitée ici est l'origine et la confiance que l'on va pouvoir accorder aux constantes utilisées. Comme on peut le voir dans ce premier exemple, un ensemble de paramètres a été sélectionné par les développeurs de la courbe Curve25519, répondant à des critères de sécurité, mais aussi de performance. Les questions qui peuvent être soulevées sont alors : quels sont ces critères ? Sont-ils publiés ? Y a-t-il une procédure transparente pour la génération des constantes ? Qui sont les développeurs derrière cette courbe ? Puis-je leur faire confiance ?

2. BRÈVE HISTOIRE DES COURBES ELLIPTIQUES

La cryptographie à base de courbes elliptiques a été introduite en 1985 par Koblitz et, de manière indépendante, par Miller. Au début des années 1990, ECDSA voit le jour (la variante de DSA utilisant les courbes elliptiques). Quand bien même de vives critiques existent au sujet des courbes elliptiques (des personnalités de la cryptographie à cette époque qualifient ce domaine d'ésotérique), l'ANSI (*American National Standards Institute*), avec l'appui de la NSA, approuve les premiers standards à la fin de l'année

1995. Le NIST publiera par la suite, en 1999, un ensemble de courbes destinées à la cryptographie. De l'ensemble de ces courbes, mises au point essentiellement par Jerry Solinas et quelques autres mathématiciens de la NSA, on retient surtout aujourd'hui celles encore considérées comme sûres d'un point de vue mathématique, c'est-à-dire celles définies sur un corps premier tel que P-224, P-256, P-384 (où le nombre situé après le P correspond à la taille du nombre premier en bits). Dans ce même temps, quelques résultats mathématiques sont obtenus, identifiant notamment des familles de courbes plus faibles en termes de sécurité. La NSA annonce en 2005 la « suite B », après notamment un accord commercial avec Certicom qui possédait alors quantité de brevets sur le domaine. Cette « suite B » est un ensemble de recommandations cryptographiques visant notamment la protection des communications classées « Top Secret » par le gouvernement états-unien, reposant à ce moment-là uniquement sur des primitives utilisant les courbes elliptiques. Cette même année verra également arriver une implémentation de quelques primitives à base d'ECC (*Elliptic Curve Cryptography*) dans la célèbre bibliothèque OpenSSL ainsi que l'introduction de courbes issues d'un développement en milieu universitaire, notamment la Curve25519 de Daniel Bernstein. Milieu universitaire qui va également mettre au point les années suivantes de nouvelles familles de courbes aux propriétés intéressantes en termes de performance et de résistance aux attaques par canaux auxiliaires, telles que la famille des courbes d'Edwards, qui seront introduites en cryptographie par Daniel Bernstein et Tanja Lange.

3. LES CRITÈRES DE SÉLECTION CONCERNANT LA SÉCURITÉ

Les deux sous-parties suivantes visent à introduire quelques éléments nécessaires à la sécurité d'une courbe elliptique. Bien entendu, l'objet de ce travail vise essentiellement à définir des critères mathématiques pour la structure d'une courbe : c'est-à-dire rendre difficile le calcul du logarithme discret dans le groupe de la courbe. Cependant, il existe d'autres critères de sécurité ayant été mis en exergue par des attaques réelles, tels que la sécurité de la courbe tordue (« twisted curve ») ou encore la nécessité d'une loi d'addition unifiée, pour ne citer que ces deux exemples.

Mais ce n'est pas tout. Il y a également des critères comme la simplicité d'implémentation d'une courbe (et donc des opérations associées) qui sont d'une importance suffisante pour considérer cet aspect comme un élément de sécurité. Sur ce dernier point, afin d'appuyer l'argument de simplicité, on pourra prendre en exemple des implémentations largement répandues qui ont été cassées à cause d'une mauvaise implémentation, notamment à cause d'une difficulté supplémentaire rajoutée par certaines courbes. Par exemple, l'implémentation de ECDH pour TLS dans le cas de Bouncy Castle a été cassée à cause d'une étape de vérification manquante (si un point utilisé était effectivement sur la courbe ou non) ayant permis d'effectuer une attaque en utilisant une courbe invalide. Cette étape de vérification de la présence d'un point sur une courbe peut paraître triviale, mais ajoute en réalité un niveau de complexité qui pourrait être réduit si quelques contre-mesures simples étaient prises en amont dans la structure de la courbe elle-même.

Cette partie n'a pas pour objectif de lister de manière exhaustive les critères de sécurité nécessaires, mais simplement d'introduire quelques notions. Pour plus de détails, vous pouvez consulter différentes publications récentes qui tentent, elles, de définir de manière relativement complète les besoins en matière de structure mathématique afin qu'une courbe soit considérée comme sûre [3][6][7].

3.1 Difficulté de résolution du logarithme discret

Bien que le calcul du logarithme discret soit considéré comme difficile pour une grande partie des courbes, il est nécessaire d'établir quelques critères afin de ne pas tomber dans des classes de courbes vulnérables à des attaques permettant l'utilisation d'un algorithme avec une complexité moindre. Le critère de base étant que la meilleure méthode disponible pour calculer le logarithme discret n'ait pas une complexité inférieure

à une certaine borne fixée, correspondant à la complexité du calcul dans un cas « classique » (c'est-à-dire une courbe n'ayant pas de faiblesse). Pour ce faire, plusieurs critères sont établis sur différents éléments de la structure de la courbe elliptique (nombre de points de la courbe, ordre du sous-groupe, degré de plongement) afin de garder une telle complexité. Pour plus de détails, le site SafesCurve donne de bonnes explications dans sa catégorie « ECDLP security » **[8]**.

Un élément supplémentaire visant à se protéger de futures attaques non connues contre le logarithme discret est l'utilisation d'une courbe générique, c'est-à-dire ne reposant par exemple pas sur un corps de base trop structuré. Pour la courbe NIST-P256, le nombre premier utilisé est de la forme $2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$, ce qui permet, dans le cas d'un groupe multiplicatif, une nette optimisation du calcul du logarithme discret. Certains cryptologues proposent donc de ne pas utiliser une telle forme de nombre premier pour les courbes elliptiques dans la crainte de la découverte d'une attaque identique contre ces dernières. On notera qu'il s'agit ici d'un compromis à faire entre la performance (les nombres premiers construits de cette dernière manière permettant d'optimiser les calculs) et la généricité de la courbe.

3.2 Sécurité des implémentations

Même si les critères précédents sont respectés, il est toujours possible d'attaquer l'implémentation même avec une possibilité de succès. Par exemple, il existe de nombreuses attaques par canaux auxiliaires (analyse des temps de calcul, injection de fautes) contre la méthode de multiplication d'un point par un scalaire (qui fait partie du cœur des opérations sur une courbe elliptique). Pour ne citer que le cas d'OpenSSL, son implémentation de l'échelle de Montgomery (algorithme de multiplication par un scalaire) a été l'objet d'une attaque par un canal auxiliaire « FLUSH+RELOAD » permettant l'extraction des « nonces » utilisés par ECDSA **[11]**. Bien d'autres problèmes potentiels existent concernant la structure de la courbe : sécurité de la courbe tordue, petit sous-groupe, points spéciaux, nécessité d'avoir une loi d'addition unifiée ; il existe dans tous les cas des contre-mesures efficaces. Pour plus de détails pratiques une fois de plus, consultez le site SafesCurve dans sa catégorie « ECC security » **[8]**.

4. LES CRITÈRES DE SÉLECTION EN TERMES DE CONFIANCE

Comme vu précédemment, les critères de sélection concernant la sécurité des courbes elliptiques, allant de la difficulté de calcul du logarithme discret à la résistance aux attaques par canaux auxiliaires, font l'objet d'un accord relativement général de la part de la communauté. Cependant, un nouveau critère est apparu ces dernières années concernant la procédure de génération d'une courbe. En effet, même si l'ensemble des critères de sécurité est respecté, une certaine souplesse existe pour un attaquant qui aurait par exemple connaissance d'une vulnérabilité pour une certaine classe de courbes. C'est essentiellement ce dernier point qu'essayent de résoudre les différentes initiatives qui ont vocation à rendre transparente et non manipulable la procédure de génération de la courbe.

4.1 Processus de génération des constantes

4.1.1 Des paramètres… tombés du ciel

La première façon de fournir des paramètres relatifs à une courbe est de les donner simplement sans explication. Difficile alors d'établir une confiance dans ces derniers, à moins justement d'avoir une confiance aveugle dans l'organisme ou la personne qui les a générés. Les paramètres de la courbe de l'ANSSI sont ainsi fournis par le biais de leur site ainsi que du Journal Officiel **[2]** :

NOM	FRP256v1
p	F1FD178C0B3AD58F10126DE8CE42435B3961ADBCABC8CA6DE8FCF353D86E9C03
A	F1FD178C0B3AD58F10126DE8CE42435B3961ADBCABC8CA6DE8FCF353D86E9C00
B	EE353FCA5428A9300D4ABA754A44C00FDFEC0C9AE4B1A1803075ED967B7BB73F
x(P0)	B6B3D4C356C139EB31183D4749D423958C27D2DCAF98B70164C97A2DD98F5CFF
y(P0)	6142E0F7C8B204911F9271F0F3ECEF8C2701C307E8E4C9E183115A1554062CFB
q	F1FD178C0B3AD58F10126DE8CE42435B53DC67E140D2BF941FFDD459C6D655E1
i	1

Paramètres de la courbe FRP256v1 tels que publiés au Journal Officiel le 16 octobre 2011.

Il apparaît trivial, avec cette manière de procéder, qu'une très grande marge de manœuvre soit offerte à celui qui fournit la courbe elliptique, quand bien même l'état de l'art en termes de sécurité serait respecté. Dans une publication visant à mettre en exergue la manipulabilité des procédures destinées à générer les paramètres de courbes elliptiques **[3]**, une méthode est proposée afin de générer une courbe possédant des propriétés particulières. Afin d'illustrer leurs propos, les auteurs proposent de générer une constante b (cf. équation classique d'une courbe) très structurée (cf. l'exemple ci-après) pour démontrer une attaque fictive. Ainsi, en respectant les meilleurs critères de sécurité connus, ils arrivent à obtenir après 78 minutes sur un cœur de CPU AMD une constante avec le motif « 5AFEBADA55ECC » (je vous laisse décoder le « leet speak ») :

Terminal

```
b = 0x5AFEBADA55ECC5AFEBADA55ECC5AFEBADA55ECC5AFEBADA55ECC5AFEBADA5A57
```

L'idée ici est de démontrer que, si un attaquant connaît une classe de vulnérabilité non connue sur les courbes elliptiques, il est alors simple de générer une courbe présente dans cette classe (l'idée derrière la structure du paramètre b étant ici de mimer une courbe appartenant à cette classe) tout en respectant les critères de sécurité connus. Le problème ici étant bien l'absence même de procédure permettant de vérifier la génération des paramètres.

4.1.2 Paramètres dérivés d'un « hash »

Une autre manière de fournir les paramètres d'une courbe est de les proposer sous la forme de « seed » et d'une fonction de hachage (plus quelques étapes afin de générer la constante finale). Le résultat de la fonction sera ainsi une partie de la sortie de la fonction de hachage. L'idée est donc de faire reposer la confiance sur la solidité de la fonction de hachage : il est très difficile de choisir un paramètre selon une certaine structure, car il est très difficile d'inverser la fonction de hachage. C'est notamment ce que proposent les courbes du NIST pour les paramètres de la courbe, à l'exception du nombre premier utilisé. Pour le paramètre b de la courbe P-256, la formule suivante est utilisée afin de le générer :

$$b = \sqrt{\left(-\frac{27}{SHA1(s)}\right)}$$

Avec :

Terminal

```
s = c49d360886e704936a6678e1139d26b7819f7e90
```

L'origine de cette dernière valeur ? Nul ne le sait. Ou plutôt, seuls les développeurs de ce paramètre (des mathématiciens de la NSA). On a donc remplacé, par rapport à la situation précédente sans procédure, un paramètre inexpliqué par une valeur de « seed » inexpliquée produisant ce paramètre. Cela complexifie donc légèrement la génération d'un paramètre répondant à une structure bien précise qui permettrait d'exploiter une vulnérabilité non connue, mais ne la rend en rien impossible. Pire, en termes de confiance, le processus n'étant en définitive pas transparent (on ne sait tout simplement pas d'où vient le « seed »), la procédure de génération n'est en rien améliorée. Cependant, concernant la difficulté de génération d'une constante avec une structure pouvant exploiter une vulnérabilité inconnue, le coût en est effectivement augmenté. De la même manière que précédemment, les auteurs de BADA55 **[3]** ont effectué une attaque cherchant, par la force brute, la présence d'un motif de 8 symboles hexadécimaux dans la constante b (BADA55EC) afin de simuler une vulnérabilité potentielle avec

une probabilité de 2^{-32}. Dit autrement, l'idée est de maîtriser 32 bits de la constante b. Cela représente, contrairement à l'attaque fictive précédente, une quantité de calcul bien plus élevée à effectuer avant d'obtenir un résultat dans les contraintes proposées. Il ne faut pas oublier que cette attaque n'est effectuée qu'une seule fois avant de publier ou normaliser la courbe et pourrait ne représenter aucune difficulté pour un organisme disposant de gros moyens de calcul. En témoigne d'ailleurs la réaction de Michael Scott en 1999 au moment de la publication des courbes du NIST :

Fichier

```
Consider now the possibility that one in a million of all curves have an exploitable
structure that "they" know about, but we don't... Then "they" simply generate a million
random seeds until they find one that generates one of "their" curves. Then they get
us to use them. And remember the standard paranoia assumptions apply - "they" have
computing power way beyond what we can muster. So maybe that could be 1 billion.
```

Le problème était donc déjà bien présent dans les esprits il y a maintenant dix-sept ans.

4.1.3 Paramètres dérivés de valeurs connues

Utiliser des valeurs de « seed » inconnues (quand bien même générées potentiellement de manière aléatoire) ne permet donc en rien d'améliorer la confiance dans la procédure. Plusieurs initiatives ont vu le jour afin d'utiliser des nombres provenant de constantes connues, par exemple : cos(1), pi ou sqrt(2). On parle alors de NUMSN, « Nothing Up My Sleeve Numbers ». Ces dernières valeurs étant considérées comme n'ayant pas, par construction, de propriétés cachées pouvant être exploitées. C'est notamment le cas de la procédure Brainpool **[4]** qui vise à générer les paramètres d'une courbe en utilisant une « seed » initiale basée sur la constante exp(1). Bien entendu, cette valeur de « seed » initiale ne remplissant pas les critères de sécurité nécessaires une fois passée dans le processus de génération (entre autres une fonction de hachage), cette valeur est incrémentée jusqu'à ce que les critères fixés soient effectivement atteints. La valeur du nombre premier utilisé est également générée selon cette procédure. Cette fois-ci on ne peut donc pas manipuler la « seed » utilisée. Malgré tout, rien ne garantit que la procédure entière n'ait pas été élaborée délibérément pour obtenir certaines constantes : le choix de la fonction de hachage, la taille des données d'entrées de celle-ci, la manière d'extraire une partie de la constante naturelle, la constante naturelle elle-même et bien d'autres petites étapes de la procédure. Il y a, une fois de plus, beaucoup de degrés de liberté pour celui qui élabore la procédure pour arriver à obtenir une certaine structure dans les constantes générées. Dans le numéro 8 de « Poc||GTFO », J.-P. Aumasson propose, dans un article élégamment nommé « Backdoors up my sleeve », une preuve de concept permettant de générer 1'990'656 constantes dérivées de « seed » ayant pour valeur uniquement des nombres NUMSN et en manipulant les méthodes d'encodage et décodage ainsi que le choix de la fonction de hachage. Bref, rien qui pourrait sembler étrange lors d'une analyse de la procédure. Cela démontre combien il est aisé, même en utilisant un nombre en apparence objectif, de manipuler quelques bits du paramètre résultants d'une telle procédure.

4.1.4 Paramètres dérivés d'une source d'aléa publique

Plus récemment, une initiative visant à utiliser une source d'aléa publique a vu le jour : « Million Dollar Curve ». L'objet intéressant de la procédure ici est d'utiliser les résultats de plusieurs loteries dans le monde afin d'en extraire des « seeds » pour générer ensuite les constantes d'une courbe. Cela réduit encore plus les degrés de liberté pour celui qui souhaiterait mettre au point une procédure biaisée en se basant sur une source d'aléa publique sans avoir à modifier directement les critères de sécurité. Cette différence complique hautement la tâche de l'attaquant et rendrait la procédure nettement plus alambiquée s'il essayait de la biaiser. Bien entendu, cette procédure repose largement sur l'hypothèse qu'un attaquant disposant de grosses capacités ne puisse influencer les résultats des différentes loteries utilisés par « Million Dollar Curve », ce qui est notamment critiqué par certains cryptologues.

4.1.5 Paramètres générés de manière déterministe

Une tout autre manière de générer les constantes est de proposer une procédure entièrement déterministe qui ne repose que sur des contraintes de sécurité, mais également de performance. L'idée est de minimiser le choix dans les constantes disponibles. C'est par exemple le cas des courbes proposées par Microsoft. Ces courbes, nommées « NUMS Curves » (cela n'est pas sans rappeler les NUMSN), utilisent par exemple cette procédure afin de générer un nombre premier :

- on définit un niveau de sécurité s (par exemple 128, 192 ou 256) ;
- on prend un nombre premier p de la forme $2^{2s} - c$ (cette structure de nombre premier permettant d'effectuer des calculs plus rapidement) ;
- on cherche c le plus petit entier tel que p 3 mod 4 (choix assez courant permettant d'optimiser certains calculs, tels que celui de la racine carrée, utile dans la compression de point).

Dans ce simple cas également, plusieurs critères sont manipulables afin de structurer le nombre premier qui sera choisi. La procédure n'apporte en définitive que les raisons pour lesquelles Microsoft a choisi précisément ces courbes. La confiance ici est donc déplacée sur les bons choix en termes de sécurité et de performance que pourrait avoir faits Microsoft (dans le cas présent, ils n'ont choisi que des éléments basés sur des critères reconnus). Exprimée de façon plus claire encore, la confiance repose sur le fait que les ingénieurs de chez Microsoft n'ont pas délibérément sélectionné l'ensemble de ces critères afin de choisir une certaine classe vulnérable de courbe.

Quand bien même les courbes Curve25519 et Curve41417 ne prétendent pas définir de procédure permettant de générer leurs paramètres, elles apportent de manière transparente, à l'instar de celles de Microsoft, un ensemble d'explications détaillant les raisons des choix de leurs constantes (notamment concernant les besoins de sécurité et performance).

4.2 Confiance, sécurité et performance : des usages aux compromis

L'ensemble des procédures détaillé précédemment met en évidence un élément important dans la sélection d'une courbe : on ne peut pas obtenir des performances intéressantes (nécessitant une courbe plus structurée, par exemple dans la forme du nombre premier utilisée) et une courbe générée selon une procédure qui dérive ses paramètres depuis des données aléatoires (ou des NUMSN). Pareillement, on ne peut pas obtenir une courbe avec une procédure totalement transparente (telle que la courbe à un million de dollars) et une courbe avec des paramètres dédiés à optimiser certains calculs. Des compromis sont effectivement nécessaires et dépendent clairement des besoins de l'utilisateur. Besoins qui d'ailleurs nécessitent la définition de différents standards en identifiant clairement des usages, proposant par là même plus de diversité. Il existe malgré tout une certaine concurrence pour obtenir certaines propriétés de performance et de sécurité, telles que les courbes NUMS de Microsoft et les courbes Curve25519 et Curve41417, prétendant toutes à détailler de manière transparente l'ensemble des choix effectués sur leurs constantes.

5. LES PROCESSUS DE STANDARDISATION

5.1 Modèles de décision

Pour faire très court et très simple, il existe principalement deux grandes catégories de modèles de décision au sein des organismes de recommandation et de standardisation. Ceux qui visent à établir un consensus de la communauté (l'IETF parle de « rough consensus ») en fournissant un processus ouvert impliquant des personnes extérieures : liste de diffusion publiquement accessible, prise de décision par les personnes

responsables du standard ou de la recommandation en fonction des discussions apportées par la communauté. C'est par exemple le cas de l'IETF ou du W3C. D'un autre côté, il y a les organismes fermés (pouvant malgré tout faire intervenir des personnes extérieures) prenant des décisions en interne auxquelles on ne peut pas participer. On pourra citer par exemple le NIST ou l'ANSSI, dont l'objet dépasse dans les deux cas l'élaboration de standards ou de recommandations.

5.2 Quoi de neuf au NIST ?

Dans une récente publication **[9]**, Daniel Bernstein et Tanja Lange font une analyse des échecs des premières courbes du NIST. Ils mettent en exergue entre autres :

- une complexité d'implémentation ;
- l'absence d'explication dans les choix des paramètres ;
- de mauvais choix d'optimisation.

Malgré tout, le NIST est en passe de développer un nouveau standard afin de proposer de nouvelles courbes (même si ce n'est pas encore sûr pour le moment). C'est ainsi qu'il a organisé un workshop durant le mois de juin 2015 **[10]**. On remarquera que l'ANSSI était présente avec deux de ses chercheurs, qui ont apporté d'ailleurs une pièce intéressante **[7]** aux débats : « Diversity and Transparency for ECC ». En effet, malgré que l'ANSSI n'ait pas apporté de justification aux paramètres de sa courbe par le passé, il semble, avec cette contribution, que la transparence et la vérifiabilité du processus de génération des constantes soient désormais d'un intérêt important.

5.3 IRTF/IETF : RFC 7748, « Elliptic Curves for Security »

Bien que pour la cryptographie l'IETF (et surtout le Crypto Forum Research Group) dépende largement des choix faits par le NIST (une bonne partie des primitives conseillées proviennent du NIST), le groupe de travail sur TLS a demandé au CFRG de lui recommander de nouvelles courbes à utiliser. Une des motivations principales pour cette demande était la nécessité d'avoir un processus de génération de courbe auquel on puisse effectivement faire confiance. Il y a eu également beaucoup de discussions sur l'intérêt des courbes dites « aléatoires » (c'est-à-dire dont les paramètres sont générés selon une procédure faisant intervenir à la base une valeur aléatoire) vis-à-vis des courbes avec des paramètres « optimisés ». La conclusion de ce débat met en avant les courbes « optimisées » avec plusieurs tailles pour les longueurs de clefs (notamment pour la crainte d'attaques futures, il est simplement proposé d'utiliser une taille de clef plus grande). C'est ainsi que les courbes candidates les plus discutées étaient celles de Microsoft (les fameuses « NUMS ») et de Daniel Bernstein (Curve25519) pour, en définitive, recommander Curve25519 (pour un niveau de sécurité de 128 bits) et la courbe Ed448-Goldilocks (une courbe d'Edwards avec des choix conservateurs où l'ensemble des paramètres est justifié) développée par Mike Hamburg (pour un niveau de sécurité de 224 bits). Pour le moment, cela ne concerne que les primitives ECDHE et ECDH (des travaux sur les signatures sont en cours), et le RFC concernant leurs usages dans TLS est en préparation.

5.4 W3C : Web Cryptography API

On pourra également remarquer que du côté du W3C, la toute récente API « web cryptography » encore en normalisation ne propose pour le moment que les vieilles courbes du NIST malgré de longues discussions sur la possible inclusion des courbes NUMS de Microsoft et la courbe Curve25519. En effet, les différents problèmes d'intégration dans la norme ainsi que l'absence

de recommandation du CFRG vis-à-vis des courbes à sélectionner (au moment du débat au W3C, car depuis le RFC a été publié) n'ont pas permis de créer le consensus nécessaire pour intégrer d'autres courbes que celles du NIST **[12]**.

CONCLUSION

Ainsi, même si les courbes elliptiques représentent une avancée intéressante en termes de sécurité, de performance et comme une alternative à RSA, il apparaît aujourd'hui primordial d'avoir des processus de sélection totalement transparents et, idéalement, ouverts. Même si, comme souvent, il n'y a pas de solution parfaite, plusieurs courbes peuvent être aujourd'hui considérées comme ayant des paramètres élaborés de façon transparente (que cela soit de manière aléatoire ou déterministe) même si les procédures sous-jacentes pour les générer (Brainpool, NUMS) restent faillibles. Aussi, les compromis à faire sont encore discutés quant aux courbes à utiliser (y a-t-il un vrai intérêt à utiliser les courbes « aléatoires » ? Même en second choix ? Pourquoi ne pas plutôt choisir une solution secondaire telle qu'une primitive « post-quantique » si l'on néglige les performances ?) et devrait certainement s'atténuer quand le NIST aura pris une décision par rapport à de potentielles prochaines recommandations. Pour l'heure, l'IETF a déjà fait ses choix pour la future version de TLS et la transparence est au rendez-vous. ▮

REMERCIEMENTS

Bien évidemment, cet article est essentiellement basé sur les différents travaux de recherche référencés dans les notes et j'en remercie sincèrement leurs auteurs. Merci également à Éloi Vanderbeken et Alexander pour leurs relectures et conseils avisés. Merci aussi à Aurélien pour ses nombreuses relectures.

RÉFÉRENCES

[1] **https://cr.yp.to/ecdh.html**

[2] **http://www.legifrance.gouv.fr/affichTexte.do;jsessionid=?cidTexte=JORFTEXT000024668816&dateTexte=&oldAction=rechJO&categorieLien=id**

[3] **https://bada55.cr.yp.to/**

[4] **https://tools.ietf.org/html/rfc5639**

[5] **https://cryptoexperts.github.io/million-dollar-curve/**

[6] **https://eprint.iacr.org/2014/130.pdf**

[7] **https://eprint.iacr.org/2015/659.pdf**

[8] **http://safecuves.cr.yp.to/**

[9] **https://cr.yp.to/newelliptic/nistecc-20160106.pdf**

[10] **http://www.nist.gov/itl/csd/ct/ecc-workshop.cfm**

[11] **https://eprint.iacr.org/2014/140.pdf**

[12] **https://lists.w3.org/Archives/Public/public-webcrypto/2014Aug/0186.html**

AND YOU MAKE CRYPTO, AND I MAKE CRYPTO, EVERYBODY'S MAKING CRYPTO !

Virginie GALINDO

Le basculement de nos vies vers le numérique est en route. Nos habitudes sociales, productives, d'achat, passent de plus en plus souvent par la case électronique. Les conversations qui avaient lieu à voix basse devant la machine à café ou de vive voix dans notre salle à manger, à l'abri des oreilles indiscrètes, se retrouvent transportées et stockées sur des serveurs et des machines que nous ne contrôlons pas toujours (en fait, jamais). La nécessité de protéger les échanges n'est pas nouvelle. Mais aujourd'hui, nous ne pouvons plus utiliser le sceau de notre famille pour sceller les parchemins, ou les plis oblitérés à enveloppes opaques. Nos transactions sont des séries de 0 et 1 (le numérique), et des outils adaptés pour les protéger se sont développés. Ces outils reposent sur l'usage de la cryptographie. Nous nous pencherons dans cet article sur les possibles utilisations de la cryptographie dans le contexte du Web. En route...

1. LE PETIT RAPPEL DE CRYPTO POUR SE METTRE EN JAMBES...

Qu'est-ce que la cryptographie ? C'est une science.

Wikipédia nous dit que « *la cryptographie est une des disciplines de la cryptologie (la science du secret) s'attachant à protéger des messages (assurant confidentialité, authenticité et intégrité) en s'aidant souvent de secrets ou clés. Elle se distingue de la stéganographie qui fait passer inaperçu un message dans un autre message alors que la cryptographie rend un message inintelligible à autre que qui de droit.* »

Et si vous avez fait un peu de crypto et de mathématiques à l'école, vous savez que toute la subtilité d'un bon expert de cryptographie est de savoir jouer avec les bonnes fonctions de mathématiques. Transformer une série de données x, en une série de données y, après avoir multiplié, divisé, mis à la puissance. Les fonctions essentielles attendues de la crypto sont donc de transformer x en y, ou de retrouver x à partir de y. Ces deux opérations simples permettent de fournir des services tels que :

- chiffrer (rendre illisible son contenu au premier abord) ou déchiffrer un document ;
- signer un document ou vérifier que la signature attachée est bien celle attendue.

Ces simples opérations permettent de protéger les données stockées dans votre PC ou votre serveur, de rendre illisibles les échanges, de rendre anonymes les personnes. Il faudra également vous souvenir que la mise en place d'une solution à base de cryptographie repose sur le fait que les deux parties souhaitant communiquer se mettent d'accord sur des algorithmes à utiliser (ainsi qu'une longueur de clé) et aient distribué de part et d'autre les bonnes clés secrètes (symétriques ou asymétriques). Voilà, c'était la séquence, rattrapage de vos notions de crypto, attaquons les choses sérieuses.

2. ET LE WEB, DANS TOUT ÇA ?

2.1 HTTPS, c'est bien...

Sur le Web, vous surfez grâce à un navigateur (posé sur votre PC, tablette, smartphone, voiture, machine à laver...). Ce navigateur accède à un serveur, qui lui délivre des pages web, décrites grâce aux technologies HTML5, CSS3 et qui embarque du code JavaScript. Ces pages sont transportées sur le protocole HTTP. Votre navigateur est une application qui interprète tout ça et finit par afficher un joli site web.

Quid de la sécurité de cet échange ? La bonne nouvelle, c'est qu'il peut y avoir de la sécurité. Les messages qui transitent entre votre navigateur et le serveur sont transportés sur des trames HTTP **[1]**, un protocole permettant de formater et router les messages entre deux nœuds internet, s'appuyant du IPv4 **[2]** ou IPv6 **[3]** (selon la modernité du site). Ces protocoles sont décrits par l'Internet Engineering Task Force, le lieu de normalisation de l'Internet. Si vous fréquentez des sites web consciencieux, votre communication s'effectue en HTTPS. Notez-le « S », comme Sécurité. Ce « S » indique que le navigateur et le serveur ont mis en place une couche supplémentaire de sécurité TLS (*Transport Security Layer*), dès le début de connexion.

TLS est également défini pas l'IETF **[4]**. TLS permet entre autres de vérifier l'authenticité du site web, et de se mettre d'accord entre client et serveur que le trafic échangé sera chiffré. Il existe de nombreuses options pour mettre en place cette communication chiffrée, nous ne rentrerons pas dans les détails.

L'avantage d'utiliser HTTPS est que l'utilisateur est protégé contre l'espionnage des échanges (*eavesdropping*) ou contre le célèbre Man In The Middle (une personne indélicate reçoit et relaie les messages entre le client et le serveur, ayant accès à tous les messages, et ayant la possibilité de les altérer).

Comment TLS fonctionne ? TLS repose sur l'utilisation de clés provisionnées du côté du site web (serveur), et permet de vérifier l'authenticité du site web, sur la base du certificat qu'il présente. Qu'est-ce donc que ce certificat ? Le certificat est une série de données, délivrées et signées par une autorité de certification. Cette autorité de certification vérifie, avant de donner ce certificat, que le site web de Monsieur Dupont était bien administré par Monsieur Dupont, sur un domaine détenu par Monsieur Dupont, et que ses activités sont à peu près légales. Ce certificat contient également la clé publique qui permettra au navigateur et au serveur de négocier une session de communication sécurisée. Dans cet échange HTTPS, établi grâce à la clé du certificat, c'est souvent le navigateur qui établit l'authenticité du site web de Monsieur Dupont. Le site web de Monsieur Dupont, lui, ne met pas en place de mécanisme pour reconnaître le navigateur. Le serveur sert simplement les requêtes qu'il reçoit de la part du serveur. L'échange de données peut se faire dans un format chiffré. Donc grâce à HTTPS, le navigateur peut vérifier que le site visité est un site authentique, et communiquer dans un canal sécurisé, où les données sont transportées chiffrées.

2.2 Mais HTTPS, c'est une sécurité liée au navigateur

Bref, un navigateur et un serveur peuvent communiquer de manière sécurisée. Mais on notera ici que les sessions HTTPS sont opérées par le navigateur lui-même. La clé qui est générée entre le serveur et le navigateur est sous le contrôle du navigateur. Or, dans la mouvance d'une certaine indépendance des acteurs qui fabriquent des navigateurs et des acteurs qui offrent des services, certains fournisseurs de service souhaiteraient avoir une sécurité qui soit sous leur propre contrôle (et non pas sous le contrôle du navigateur). Par exemple, une sécurité qui puisse être intégrée directement dans ses pages web. Le site web de Monsieur Dupont, qui vend des chaussettes de luxe, pourrait ainsi établir une communication sécurisée, selon ses algorithmes et les paramètres de son choix, et échanger des données sensibles telles que la pointure de pied de Kevin ou son numéro de carte bancaire.

Mais gérer sa sécurité au niveau de son site web, est-ce possible ? Oui. Car le World Wide Web Consortium (W3C) s'en est mêlé.

2.3 Et si on veut faire de la sécurité dans les web apps, on fait comment ?

Si on veut faire de la sécurité directement dans les applications web (ou sites web), et que l'on veut qu'elle soit normalisée, on fait appel au W3C. Le W3C est un organisme qui développe le standard du web, notamment avec HTML5 et CSS3. Les standards délivrés par le W3C sont appelés des recommandations, et la procédure pour sortir un document technique est très bien définie. Le W3C mise sur la définition collective d'une capacité technique, bien souvent, avec la participation des principaux fabricants de navigateurs (Apple, Mozilla, Microsoft, Google, Yandex, Opera). Par ailleurs, le W3C, contrairement à l'IETF, propose des spécifications avec une licence de type Royalty Free. C'est-à-dire elle garantit que tout utilisateur de la spécification ne se verra pas ennuyer par un détenteur de brevet, qui aurait participé au travail d'élaboration de la spécification. Vous comprenez donc l'intérêt d'avoir un groupe de travail le plus fréquenté possible (au mieux avec tous les fabricants de navigateurs). Par ailleurs, le W3C est économe de ses ressources, si un groupe de travail se crée au W3C et est bien fréquenté, cela dénote d'un intérêt certain de l'industrie, et la spécification a toutes les chances de finir dans les navigateurs principaux.

3. MAIS REVENONS À NOTRE CRYPTO GÉRÉE PAR UN SITE WEB ?

3.1 La librairie Web Crypto

Le W3C a ouvert un groupe de travail technique sur le sujet de la cryptographie, il y a 3 ans. L'objectif de ce groupe était de développer une librairie d'outils de cryptographie, mise à la disposition des développeurs d'applications web (ou sites web), directement intégrée dans le navigateur. Le premier cas d'usage qui a réuni les acteurs du web autour d'une table, et qui a donné lieu à ce groupe de travail, était l'identité. Comment vérifier l'identité d'un utilisateur de service sur le Web ? Cette question, bien que très simple, trouve une grande variété de réponses techniques. Toutes ces techniques reposent sur des briques communes – je crois que vous me voyez venir – la cryptographie. C'est ainsi que le W3C a décidé de créer un groupe de travail technique nommé le W3C Web Cryptographic Working Group – et m'en a confié la direction. En 2012, le groupe a lancé les travaux, collectant les cas d'usage, de la part des participants et du public.

Quels cas d'usage retrouve-t-on ? La gestion de l'authentification d'un utilisateur, la protection des documents, le rapatriement de données dans le cloud, la signature de documents, l'échange de messages sécurisés, et l'usage d'objet JSON protégés (format couramment utilisé dans le Web pour transporter des données).

Afin de répondre à l'ensemble de ces scénarios, le groupe de travail a dû longuement discuter plusieurs points essentiels :

- Le niveau du service de l'API qui serait intégré dans le navigateur – plutôt des fonctions qui cacheraient la tuyauterie de la cryptographie, ou quelque chose de plus près des algorithmes...
- Les algorithmes mis justement à disposition des développeurs web – il en existe une grande variété, comment choisir.

Les exigences pour assurer l'interopérabilité entre un client navigateur et un serveur – en effet, si un seul des acteurs d'une communication chiffrée ne sait utiliser un algorithme, la communication a peu de chance d'être efficace.

3.2 Les fonctions de l'API Web Crypto du W3C

Après plus de deux années de travail intense, des milliers de mails échangés sur la liste publique, des heures passées en conférence téléphonique – parfois tard le soir, pour s'accommoder des fuseaux horaires de tous les participants venant d'Asie, d'Amérique du Nord et d'Europe, le groupe de travail a enfin délivré une spécification presque stable. Si tout va bien, ce document technique devrait devenir une Recommandation du W3C. L'avantage de passer par cette case du W3C, est que cette fonctionnalité sera présente dans tous les navigateurs, et permet de garantir un fonctionnement similaire d'un navigateur à l'autre. Mais avant de rentrer dans les détails de l'usage de cette librairie, penchons-nous sur les services qu'elle offre.

3.2.1 Génération de chiffres aléatoires

La génération de chiffres aléatoires est essentielle dans la science du secret. En effet, les chiffres aléatoires sont utilisés comme paramètres de certains algorithmes (*nonce*), ou pour compléter des données qui doivent souvent être un multiple d'entiers (*padding*). Par ailleurs, les clés sont

générées à partir de chiffres aléatoires, et il s'agit d'avoir une entropie suffisante afin de ne pas régénérer trop souvent le même chiffre aléatoire – un attaquant pourrait alors rejouer la génération de nombres et deviner plus facilement les secrets qu'il a permis de générer. Donc, la libraire de Web Crypto (appelons-la par son petit nom), donne la possibilité de générer un nombre aléatoire – mais recommande de ne pas l'utiliser directement pour générer des clés, puisqu'il y a une fonction spécifique pour faire cela. À noter tout de même que la libraire Web Crypto n'a pas de contrôle sur la qualité de l'entropie du générateur de chiffre aléatoire. C'est la machine sur laquelle tourne le navigateur qui fournit le service.

3.2.2 Création d'une clé secrète

Les clés secrètes en crypto, c'est la vie. Il existe donc une fonction de création de clés. Au cours de la création de la clé, le web développeur pourra choisir le type d'algorithme qu'il souhaite utiliser avec cette clé, ainsi que sa longueur. La clé générée ne pourra être utilisée qu'avec l'algorithme pour lequel elle a été créée. Dans le cas d'un algorithme de type clé publique/clé privée, c'est une paire de clés qui sera générée. À noter que toutes les clés générées par une application web sont temporaires, et valables au cours de la session de l'application. En effet, la plateforme web est incapable de garantir qu'une clé stockée dans la mémoire de la machine qui l'accueille soit pérenne dans le temps. Par ailleurs, le stockage de la clé est sous la responsabilité du navigateur, et laissé libre au choix du navigateur.

3.2.3 La manipulation de clés : importer, export, wrap, unwrap

Il arrive que les opérations de chiffrement ou signature doivent se faire avec une clé provenant d'une autre application, ou d'un serveur. Il existe donc une fonction pour importer ou exporter des clés. Dans ce cas, le message est encrypté (autant être prudent dans l'échange de messages sensibles). La clé est donc enveloppée (wrappée) dans un format particulier, encryptée de manière particulière. Il existe donc une fonction qui permette de wrapper ou unwrapper cette clé.

Tous les formats de clé ne sont pas supportés dans la manipulation des clés. Dans les fonctions citées, les clés ne pourront être manipulées que sous des formats dits *raw* (ensemble d'octets, adapté pour le transport de clé symétrique), *public key*, *private key* ou *JSON Web Key* (ou JWK, c'est-à-dire le format JSON adapté à l'encapsulation des clés).

3.2.4 Génération de signatures et chiffrement des données

Ces fonctions sont très simples, on fournit en paramètre les données à signer ou chiffrer, on indique la clé et l'algorithme à utiliser et le résultat sort sous forme d'une série de données chiffrées ou d'une signature.

Pour les détails du code, le lecteur pourra se référer à la spécification de cette librairie, disponible sur le site du W3C : **https://www.w3.org/TR/WebCryptoAPI/**. À sa lecture, on pourra noter que cette spécification est à destination en premier lieu des fabricants de navigateurs. Elle contient quelques exemples de code pour les développeurs d'applications web (les « users » dans la spécification).

3.3 Les algorithmes offerts par l'API

Le choix des algorithmes supportés par ces fonctions de chiffrement et génération de signature est évidemment très important. Le groupe du W3C a longtemps discuté des mérites des algorithmes qu'il fallait référencer dans cette spécification. En effet, il s'agissait de référencer des algorithmes présentant les qualités suivantes : i) être normalisés (c'est-à-dire bien définis et connus) ii)

avoir une robustesse éprouvée (bref, avoir un bon CV, et ne pas être réputé pour être cassable facilement), iii) être répandu, implémenté dans de nombreux équipements. Le choix a donc été conduit par la présence de ces algorithmes dans les plateformes d'exécution qui accueillent les navigateurs, tous environnements confondus (PC, tablette, smartphone...). La liste exhaustive des algorithmes présents dans ces environnements, a ensuite été réduite pour en extraire un minimum commun presque commun. La spécification décrit une série d'algorithmes, et la façon dont un navigateur devra les implémenter.

Pour les fanas de crypto, voici la liste des algorithmes décrits dans le document : RSASSA-PKCS1 v1.5, RSA-PSS, RSA-OAEP, ECDSA, ECDH, AES-CTR, AES-CBC, AES-CMAC, AES-GCM, AES-CFB, AES-KW, HMAC, DH, SHA-1, SHA-256, SHA-384, SHA-512, CONCAT, HKDF-CTR, PBKDF2.

Néanmoins, tous les équipements ne supportent pas tous l'intégralité de ces algorithmes, c'est pour cette raison que la spécification a extrait une série d'algorithmes recommandés pour les développeurs web. Ces algorithmes sont les plus répandus et réputés pour leur pérennité (aka, pas encore suspect pour leur faiblesse).

Pour la génération de signatures : HMAC à base de SHA-1, HMAC à base de SHA-256, RSA-PSS avec SHA-256 et MGF1 avec SHA-256, ECDSA implémenté avec la famille P-256 curve et SHA-256.

Pour le chiffrement et le déchiffrement : RSASSA-PKCS1-v1_5 associé à SHA-1, RSA-OAEP associé à SHA-256 et MGF1 avec SHA-256, AES-CBC.

3.4 L'état de la spécification Web Crypto

Le W3C développe ses spécifications techniques selon une procédure bien établie, détaillée dans un très long document. Le principe de la procédure est la suivante : une spécification est d'abord à l'état de brouillon, suffisamment étoffée et stable pour être soumise aux commentaires du public (*First Public Working Draft*), puis selon les commentaires des contributeurs, du grand public, le document est revu, corrigé, les bugs sont traités, et le document passe par plusieurs itérations de *Working Draft*. Puis, lorsque le travail a avancé au point qu'au moins deux implémentations interopérables ont été démontrées, la spécification peut devenir une *Candidate Recommendation*. C'est à ce moment que la magie de la licence gratuite opère. Au moment où la spécification devient *Candidate Recommendation*, les membres du W3C ont 60 jours pour confirmer ou non leur intention de partager avec le monde du Web leurs brevets à titre gracieux. Passés ces 60 jours, sans déclaration d'exclusion d'aucun membre, les implémenteurs de la spécification ont la garantie qu'aucune licence de brevet ne leur sera réclamée par les membres du W3C ayant participé aux travaux dans le groupe de travail. Suite à cette période, les commentaires de la communauté du web sont recueillis et traités. Techniquement affinée, garantie sans brevet, la spécification est enfin prête pour être approuvée par les membres du W3C ainsi que son directeur, Tim Berners Lee, l'inventeur du Web. Elle devient une Recommandation officielle du W3C. C'est la consécration. On récapitule, donc. *First Public Working Draft*, puis *Working Draft*, puis *Candidate Recommendation* (et appel à exclusion des brevets), puis enfin *Proposed Recommendation* (le directeur approuve), et enfin, *Recommendation*.

À l'heure où ces lignes sont écrites (tard dans la nuit, donc), la spécification a dépassé le stade de *Candidate Recommendation*. Des premières implémentations ont été démontrées sur Chrome, Safari, IE 11 et Firefox. Une table complète des implémentations et compatibilités peut être suivie sous **[5]**. Par ailleurs, une première utilisation a été documentée par Netflix, qui souhaite diffuser ses contenus en streaming directement en HTML5, tout en bénéficiant de la sécurité des flux **[6]**. Enfin, il existe des premières librairies construites sur cette API Web Crypto, comme par exemple une version Node.js de cette API **[7]** ou encore une version dédiée à l'usage des clés publiques/privées **[8]**.

À ce jour, la spécification ne peut avancer plus loin puisque les implémentations des principaux navigateurs présentent des incohérences. Tous les algorithmes ne sont pas présents dans toutes les plateformes, et tous les formats de clé ne sont pas supportés par tous. Le groupe de travail se concentre donc en ce moment sur une harmonisation des usages, et encourage les fabricants de navigateurs à aligner leurs choix.

3.5 L'API Web Crypto est-elle 100% sécurisée ?

Aucun expert de la sécurité ne garantira jamais une sécurité à 100%. Le postulat de base est assez simple, en sécurité : la sécurité est une affaire de puissance de calcul. Plus on dispose de moyens de calcul importants, plus il est facile de violer un système protégé. Deuxième postulat, un système que l'on cherche à protéger aura comme niveau de robustesse le point le plus faible de son ensemble. Vous pouvez ainsi rendre très robuste un algorithme mathématique, mais si la clé secrète qui le fait tourner est à la portée de tout le monde, le système est rapidement compromis. Le Web est par ailleurs un terrain particulier au regard des postulats précédents. Une page web n'a pas vraiment de frontière définie, puisqu'elle peut embarquer un appel à du code provenant de différentes ressources (serveurs amis, ou tierces parties). Par ailleurs, le navigateur repose sur un environnement d'exécution que le navigateur ne contrôle pas toujours, et qui peut être éventuellement compromis. Ainsi, les pages web peuvent embarquer du code malicieux, ou la gestion de la mémoire du navigateur peut être compromise, laissant un attaquant utiliser une clé de manière non autorisée. Bref, on l'aura compris, dans le cas du Web, le système à protéger est très large, soumis à des points d'entrées nombreux, et les vecteurs d'attaque sont nombreux. La sécurité fournie par l'API Web Crypto est une protection supplémentaire contre les attaques logicielles, et permet d'augmenter un peu la difficulté et le coût d'attaque sur des contenus web. En aucun cas, l'API Web Crypto ne pourra protéger les applications web des attaques locales, faites par le biais d'une machine compromise. Dans tous les cas, cette API Web Crypto doit donc être utilisée aussi dans un contexte sécurisé, c'est-à-dire avec au moins un transport de type HTTPS, et si possible en ayant mis en place CORS (C*ross-Origin Ressource Sharing*) et CSP (*Content Security Policy*), qui permettent de contrôler la bonne santé des ressources qu'une page web ira chercher pour exécuter du code provenant de sites web partenaires.

3.6 Peut-on faire mieux ?

L'API Web Crypto ne peut pas sauver le Web à elle toute seule. Néanmoins elle présente quelques vertus, et il existe des pistes d'améliorations pour renforcer son implémentation et démocratiser son usage. Ces pistes d'améliorations sont dans la feuille de route du W3C, mais ne pourront être mises en pratique tant que la spécification ne sera pas une Recommandation finalisée.

Nous pouvons lister ici un certain nombre de points d'améliorations :

- La définition d'un mécanisme de découverte d'algorithme. À ce jour, le développeur web n'a pas de garantie que tel ou tel algorithme sera supporté dans tel ou tel environnement. La principale raison à cela est la disparité des implémentations dans les machines tablette ou PC.
- Le stockage des clés permanent, qui permettrait d'étendre l'usage des clés entre deux sessions d'une application web. Le stockage dans une puce sécurisée fait partie des options envisagées.
- La gestion simplifiée des certificats contenant les clés, certifiées. Aujourd'hui l'API ne laisse manipuler que des clés symétriques, ou asymétriques, mais ne permet pas de traiter les certificats, qui sont le moyen le plus courant de distribuer des clés.
- La maintenance des algorithmes est une des clés du succès d'une telle API. De nouveaux algorithmes sont en cours d'élaboration. Notamment, suite à une rumeur de faiblesses de

certains algorithmes, laissées soigneusement par des gouvernements dans le but de profiter de leur faille, les familles d'algorithmes de type « elliptic curves » ont connu un renouveau, sur la base d'opérations et de paramètres mathématiques, définis indépendamment des gouvernements incriminés. Les acteurs de l'Internet et du Web sont en train de se mettre d'accord sur une série d'algorithmes, un peu plus recommandables.

- La définition d'une API de plus haut niveau, qui ne requiert pas de la part du développeur web des connaissances poussées en cryptographie. Aujourd'hui, l'API nécessite que le développeur choisisse un algorithme, une clé, parfois des données de padding. Il doit maîtriser un certain nombre de petites opérations, dont les implications sécuritaires ne sont pas à la portée du premier développeur venu. Par ailleurs, cette nécessaire montée en compétence peut lui faire perdre du temps. Il serait donc souhaitable qu'une fois la spécification implémentée dans les navigateurs, une série de librairies de plus haut niveau correspondant aux usages les plus fréquents soit définie.

CONCLUSION

Vous avez sans doute compris que l'API Web Crypto est un outil intéressant pour les développeurs web, afin de les aider à construire leur propre modèle de sécurité, adapté à leur souhait technique. Cette API vient renforcer l'usage de la sécurité que le navigateur peut exiger, par des connexions basées sur une combinaison de HTTP et TLS. La sécurité en matière de Web mérite des évolutions et l'ensemble des acteurs du Web y travaille dur, notamment au W3C ! Les initiatives des navigateurs web sont aussi importantes en la matière. Ces acteurs ont tout intérêt à conserver la confiance des utilisateurs pour pouvoir les garder avec eux, et monétiser leurs habitudes de surf. Je ne peux donc que vous conseiller de suivre de près les blogs sécurité des équipes de Google [9] ou Mozilla [10]. ■

RÉFÉRENCES

[1] RFC pour HTTP2 : **https://tools.ietf.org/html/draft-ietf-httpbis-http2-17**

[2] RFC pour Internet Protocol version 4 : **https://tools.ietf.org/html/rfc791**

[3] RFC pour Internet Protocol version 6 : **https://tools.ietf.org/html/rfc2460**

[4] RFC pour TLS 1.2 : **https://www.rfc-editor.org/rfc/rfc5246.txt**

[5] Charles Engelke, le blog pour suivre les implémentations de Web Crypto dans les navigateurs : **http://blog.engelke.com/2015/03/06/where-can-you-use-the-web-cryptography-api-today/**

[6] Netflix : usage de la librairie Web Crypto : **http://techblog.netflix.com/2013/07/nfwebcrypto-web-cryptography-api-native.html**

[7] Librairie Web Crypto pour Node.js : **https://github.com/skjindal93/W3C-Cryptography-API-NodeJS**

[8] Plugin pour gérer des clés asymétriques : **https://github.com/GlobalSign/PKI.js**

[9] Blog de sécurité de Google : **https://googleonlinesecurity.blogspot.fr/**

[10] Blog de sécurité de Mozilla : **https://blog.mozilla.org/security/**

4

4 LES AVANCÉES D'AUJOURD'HUI ET DE DEMAIN

À découvrir dans cette partie...

4.1 Cryptographie sur les courbes elliptiques : usages, intérêts et limites

Initiez-vous aux courbes elliptiques par le biais d'applications pratiques ! p. 104

4.2 Le grand défi du post-quantique

L'ordinateur quantique est une illusion pour certains et une réalité proche pour d'autres : découvrez les solutions possibles en cryptographie pour pouvoir résister aux futures attaques que nous promet le calcul quantique. p. 116

CRYPTOGRAPHIE SUR LES COURBES ELLIPTIQUES : USAGES, INTÉRÊTS ET LIMITES

Renaud LIFCHITZ

Peu connues du grand public, elles sont pourtant omniprésentes dans nos échanges électroniques quotidiens. Quels sont les usages et caractéristiques cryptographiques intéressantes des courbes elliptiques ?

Les courbes elliptiques sont un formidable outil de cryptographie asymétrique tant elles apportent d'usages intéressants et variés. On les retrouve dans de nombreux protocoles utilisés quotidiennement sur Internet comme TLS (RFC 4492 : « Elliptic Curve Cryptography Cipher Suites for TLS »), PGP (RFC 6637 : « Elliptic Curve Cryptography in OpenPGP »), ou encore SSH (RFC 5656 : « Elliptic-curve algorithm integration in the Secure Shell transport layer »), souvent à travers l'acronyme générique ECC (« Elliptic Curve Cryptography »).

De célèbres applications ou projets comme Tor, Apple iMessage ou Bitcoin (à travers la courbe elliptique secp256k1, cf. **[1]**) en font aussi un usage central pour préserver l'authenticité des messages, leur confidentialité ou l'anonymat des parties.

Les courbes elliptiques permettent de multiples usages :

- l'échange de clé avec l'algorithme ECDH (*Elliptic Curve Diffie-Hellman*) ;
- la signature électronique avec l'algorithme ECDSA (*Elliptic Curve Digital Signature Algorithm*) ;
- le test de primalité avec l'algorithme ECPP (*Elliptic Curve Primality Proving*) ;
- la factorisation d'entiers avec l'algorithme ECM (*Elliptic Curve Method*).

Les courbes elliptiques bénéficient en plus d'une meilleure robustesse cryptographique, d'une plus faible empreinte mémoire et d'une plus grande rapidité de calcul que les cryptosystèmes asymétriques non elliptiques.

1. QU'EST-CE QU'UNE COURBE ELLIPTIQUE ?

La cryptographie sur les courbes elliptiques repose non pas sur nos classiques opérations sur les entiers, mais sur l'utilisation astucieuse d'un ensemble de points sur un plan en deux dimensions. On s'intéresse ainsi à l'ensemble des points (x, y) solutions de l'équation cubique $A.\ x^3 + B.\ x^2.\ y + C.\ x.\ y^2 + D.\ y^3 + E.\ x^2 + F.\ x.\ y + G.\ y^2 + H.\ x + I.\ y + J = 0$ pour des constantes $A, B, C, D, E, F, G, H, I, J$ fixées.

Comme cette forme ne garantit pas l'unicité d'une seule équation par courbe, elle est rarement utilisée en pratique, et en procédant à des changements de variables, on trouve les équations simplifiées de Weierstrass. En se restreignant à certaines courbes aux caractéristiques intéressantes, on obtient aussi les équations normalisées de Montgomery et d'Edwards :

Nom de l'équation	Équation
Équation longue de Weierstrass	$y^2 + a.\ x.\ y + c.\ y = x^3 + b.\ x^2 + d.\ x + e$
Équation courte de Weierstrass	$y^2 = x^3 + a.\ x + b$
Équation de Montgomery	$b.\ y^2 = x^3 + a.\ x^2 + x$
Équation d'Edwards	$x^2 + y^2 = 1 + a.\ x^2\ y^2$ ou $x^2 + y^2 = a^2\ (1 + b.\ x^2\ y^2)$

On prendra bien soin que la courbe n'ait pas de singularité (points doubles ou points de rebroussement), sinon les caractéristiques attendues ne sont pas au rendez-vous. À ce titre, il faut veiller à ce que ce qu'on appelle le discriminant de la courbe soit non nul. Pour l'équation courte de Weierstrass, le discriminant est égal à $-16.\ (4.\ a^3 + 27.\ b^2)$.

On ajoute à cet ensemble un point fictif appelé « point à l'infini », situé arbitrairement loin de la courbe, noté O. Le tout forme un groupe au sens mathématique du terme (**https://fr.wikipedia.org/wiki/Groupe_(math%C3%A9matiques)**) sur lequel on définit une opération d'addition de deux points. En effet, en traçant une droite reliant deux points quelconques sur cette courbe réelle, on intersecte toujours la courbe

en un troisième point. On prend le symétrique de ce point par rapport à l'axe des abscisses pour définir le point résultat de l'addition (voir figure 1). Par extension, l'addition d'un point avec lui-même (appelée doublement) se fait en traçant la tangente en ce point.

L'équivalent de la multiplication dans ce groupe sera l'addition de points et l'équivalent de l'élévation à la puissance, la multiplication d'un point par un entier (l'addition multiple du point avec lui-même). Ainsi, on notera *8. P* le point *P + P + P + P + P + P + P + P*. On peut calculer algébriquement le point résultat à l'aide des formules de la figure 2. Remarquons qu'on peut utiliser les principes de l'exponentiation binaire et calculer *23. P = 1. P + 2. P + 4. P + 16. P* par calcul et addition des doubles successifs utiles.

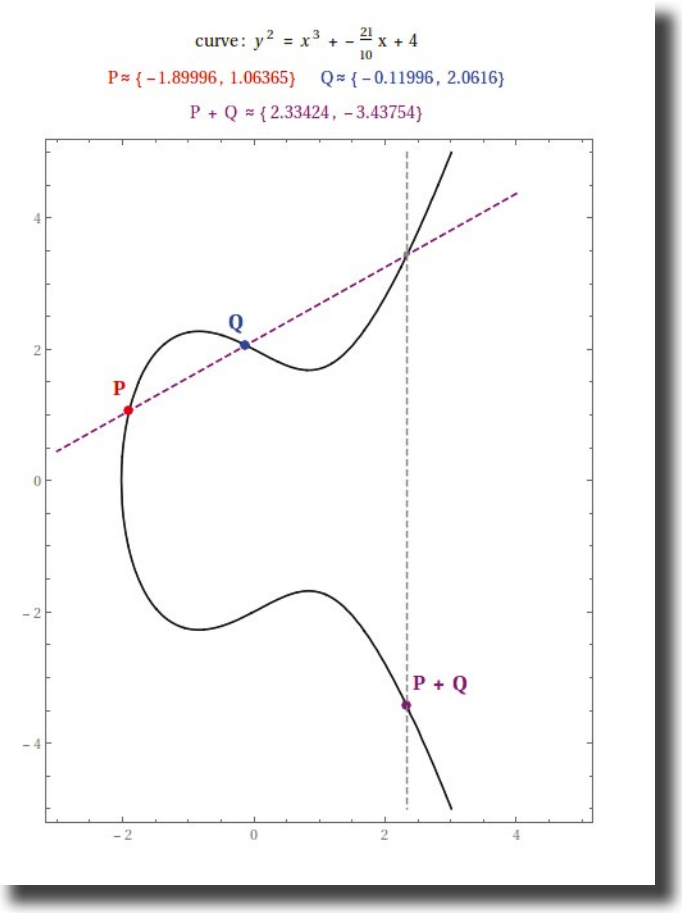


Fig. 1 : Représentation d'une courbe elliptique et addition géométrique de 2 points.

Soit $P : (x_1, y_1)$ et $Q : (x_2, y_2)$ deux points sur la courbe elliptique E d'équation $y^2 = x^3 + ax + b$ avec $4a^3 + 27b^2 \neq 0$.

$$\text{Soit } \lambda = \begin{cases} \frac{y_1 - y_2}{x_1 - x_2} & \text{si } x_1 \neq x_2 \\ \frac{3x_1^2 + a}{2y_1} & \text{si } x_1 = x_2 \end{cases}$$

alors $R = P + Q : (x_3, y_3)$ est défini par :

$$\begin{cases} x_3 = \lambda^2 - x_1 - x_2 \\ y_3 = \lambda(x_3 - x_1) + y_1 \end{cases}$$

Fig. 2 : Addition algébrique de 2 points sur une courbe elliptique.

Il est bien plus intéressant de considérer maintenant l'équation de la courbe elliptique modulo un nombre premier *p*, et de retenir donc uniquement les solutions *(x, y)* modulo *p*. Pourquoi ? Car le nombre d'éléments d'un tel groupe est bien plus diversifié que celui d'un groupe multiplicatif comme *Z / p Z* avec *p* premier (toujours égal à *p – 1*), ce qui nous assure des propriétés très intéressantes. Le nombre d'éléments de ce groupe est d'ailleurs appelé l'ordre de la courbe.

De nombreuses courbes elliptiques ont été étudiées et certaines ont d'ailleurs été normalisées dans des standards, comme celui du NIST américain (*National Institute of Standards and Technology*) ou du SECG (*Standards for Efficient Cryptography Group*). OpenSSL possède d'ailleurs une base de données de quelques courbes répandues, avec lesquelles il sait calculer :

Terminal

```
$ openssl ecparam -list_curves
  secp112r1 : SECG/WTLS curve over a 112 bit prime field
  secp112r2 : SECG curve over a 112 bit prime field
  secp128r1 : SECG curve over a 128 bit prime field
  secp128r2 : SECG curve over a 128 bit prime field
  secp160k1 : SECG curve over a 160 bit prime field
  secp160r1 : SECG curve over a 160 bit prime field
  secp160r2 : SECG/WTLS curve over a 160 bit prime field
  secp192k1 : SECG curve over a 192 bit prime field
  secp224k1 : SECG curve over a 224 bit prime field
  secp224r1 : NIST/SECG curve over a 224 bit prime field
  secp256k1 : SECG curve over a 256 bit prime field
  secp384r1 : NIST/SECG curve over a 384 bit prime field
  secp521r1 : NIST/SECG curve over a 521 bit prime field
  prime192v1: NIST/X9.62/SECG curve over a 192 bit prime field
(...)
```

Et il est bien évidemment possible d'obtenir tous les paramètres d'une de ces courbes, comme par exemple « secp256k1 », la courbe elliptique utilisée par Bitcoin :

Terminal

```
$ openssl ecparam -param_enc explicit -conv_form uncompressed -text -noout
-no_seed -name secp256k1
Field Type: prime-field
Prime:
    00:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:
    ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:fe:ff:
    ff:fc:2f
A:    0
B:    7 (0x7)
Generator (uncompressed):
    04:79:be:66:7e:f9:dc:bb:ac:55:a0:62:95:ce:87:
    0b:07:02:9b:fc:db:2d:ce:28:d9:59:f2:81:5b:16:
    f8:17:98:48:3a:da:77:26:a3:c4:65:5d:a4:fb:fc:
    0e:11:08:a8:fd:17:b4:48:a6:85:54:19:9c:47:d0:
    8f:fb:10:d4:b8
Order:
    00:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:
    ff:fe:ba:ae:dc:e6:af:48:a0:3b:bf:d2:5e:8c:d0:
    36:41:41
Cofactor:  1 (0x1)
```

où l'on retrouve respectivement le nombre premier *p*, les paramètres *a* et *b* de l'équation courte de Weierstrass, les coordonnées *(x, y)* d'un générateur (un point de la courbe qui génère tous les autres par addition successive), et l'ordre de la courbe.

Pour manipuler des courbes elliptiques simplement, nous utiliserons par la suite l'excellent logiciel open source Sage (cf. [2]), interpréteur Python interactif incluant de nombreuses fonctions mathématiques :

Terminal

```
$ sage
 ┌────────────────────────────────────────────────────────────────────┐
 │ SageMath Version 7.0, Release Date: 2016-01-19                     │
 │ Type "notebook()" for the browser-based notebook interface.        │
 │ Type "help()" for help.                                            │
 └────────────────────────────────────────────────────────────────────┘
```

```
sage: a, b, c, d, e = var('a b c d e')

sage: EllipticCurve([a, b, c, d, e])    # forme longue de Weierstrass
Elliptic Curve defined by y^2 + a*x*y + c*y = x^3 + b*x^2 + d*x + e over
Symbolic Ring

sage: EllipticCurve([a, b])    # forme courte de Weierstrass
Elliptic Curve defined by y^2 = x^3 + a*x + b over Symbolic Ring
```

Si nous essayons de travailler sur une courbe elliptique singulière (i.e. de discriminant nul), Sage lève une exception :

Terminal

```
sage: E = EllipticCurve([-3, 2])
---------------------------------------------------------------------------
ArithmeticError                           Traceback (most recent call last)
<ipython-input-38-2429dcff513f> in <module>()
(...)

ArithmeticError: invariants (0, 0, 0, -3, 2) define a singular curve
```

Pour travailler sur une courbe elliptique modulo un nombre premier, il suffit d'insérer en premier argument la définition du corps premier désiré :

Terminal

```
sage: E = EllipticCurve(GF(31),[2, 3]); E
Elliptic Curve defined by y^2 = x^3 + 2*x + 3 over Finite Field of size 31
```

Le nombre de points sur une courbe elliptique modulo p semble erratique, mais est loin d'être aléatoire. En effet, le théorème de Hasse (cf. **[3]**) précise qu'il est strictement compris entre $p + 1 - 2\sqrt{p}$ et $p + 1 + 2\sqrt{p}$, quel que soit p.

Il n'est pas nécessaire de compter les points un à un pour déterminer l'ordre exact de la courbe elliptique. Il existe un algorithme très efficace, l'algorithme de Schoof (cf. **[4]**), dont le fonctionnement interne est complexe, mais qui revient à calculer l'ordre e de la courbe modulo de petits nombres premiers (i.e. e modulo 2, e modulo 3, e modulo 5...), jusqu'à ce qu'il ne reste plus qu'une seule possibilité dans l'intervalle prévu par le théorème de Hasse, puis de trouver quelle est cette possibilité grâce au théorème des restes chinois (cf. **[5]**). Une fois calculé, cet ordre est mis en cache et tout appel futur est quasi-immédiat :

Terminal

```
sage: p = random_prime(2^256); E = EllipticCurve(GF(p), [1, 1]); E
Elliptic Curve defined by y^2 = x^3 + x + 1 over Finite Field of size 73244546
669291830923964861882895530691163756857071141401913005079061489218489

sage: time E.order()
CPU times: user 11.2 s, sys: 0 ns, total: 11.2 s
Wall time: 11.2 s
732445466692918309239648618828955306914331000555418290959805832598439874034 46

sage: time E.order()
CPU times: user 0 ns, sys: 0 ns, total: 0 ns
Wall time: 11 µs
732445466692918309239648618828955306914331000555418290959805832598439874034 46
```

Si on se fixe un point *P* de la courbe elliptique, l'ensemble de ses multiples possibles forme un sous-groupe de la courbe elliptique. Le théorème de Lagrange (cf. **[6]**) nous indique alors que la taille de ce sous-groupe (aussi appelé ordre du point) divise toujours l'ordre de la courbe elliptique correspondante :

Terminal

```
sage: E = EllipticCurve(GF(31),[1,2]) ; E
Elliptic Curve defined by y^2 = x^3 + x + 2 over Finite Field of size 31
sage: E.order()
24
sage: print [P.order() for P in E.points()]
[1, 12, 12, 4, 4, 12, 12, 6, 6, 24, 24, 3, 3, 24, 24, 24, 24, 24, 24, 8, 8, 8, 8, 2]
```

Si l'ordre de la courbe est lui-même un nombre premier, tous les ordres des points sont alors égaux à l'ordre de la courbe (sauf pour le point *O*, élément neutre de la courbe) :

Terminal

```
sage: E = EllipticCurve(GF(31),[1,3]) ; E
Elliptic Curve defined by y^2 = x^3 + x + 3 over Finite Field of size 31
sage: E.order()
41
sage: print [P.order() for P in E.points()]
[1, 41, 41, 41, 41, 41, 41, 41, 41, 41, 41, 41, 41, 41, 41, 41, 41, 41,
41, 41, 41, 41, 41, 41, 41, 41, 41, 41, 41, 41, 41, 41, 41, 41, 41, 41,
41, 41, 41, 41, 41]
```

Un point dont l'ordre est égal à l'ordre de la courbe est appelé « générateur » de la courbe, car l'ensemble des sommes du point avec lui-même génère tous les autres points de la courbe. Dans ce dernier exemple, tous les points (sauf l'élément neutre) sont générateurs, ce qui est particulièrement utile en cryptographie asymétrique, car chaque point de la courbe peut servir de base pour générer une clé publique, ce qui assure un espace de clé maximal.

2. INTÉRÊTS NOTABLES DES COURBES ELLIPTIQUES UTILISÉES EN CRYPTOGRAPHIE

2.1 Un outil de factorisation hors du commun

Les courbes elliptiques fournissent un algorithme de factorisation particulièrement efficace pour trouver un facteur de petite taille dans un grand entier, avec un temps de calcul dépendant essentiellement de la taille du facteur recherché et non de l'entier dans lequel on le cherche, à l'opposé des méthodes basées sur les cribles à corps de nombres (de type NFS). Le fonctionnement de cet algorithme, appelé ECM, est décrit dans un précédent hors-série de *MISC* (cf. **[7]**).

L'outil `socat` a fait l'objet d'une récente vulnérabilité, introduite lors d'une mise à jour de sa clé Diffie-Hellman à 1024 bits en vue d'une conformité FIPS, en janvier 2015 (cf. **[8]**). En effet, cette clé, censée être un nombre premier, s'avère être un nombre composé. L'algorithme ECM implémenté dans Sage nous permet d'exhiber facilement un facteur de ce nombre :

Terminal

```
sage: n=int('CC17F2DC96DF59A446C53E0EB826550CE388C1CEA7BCB3B
F1694D8A945A2CEA95B22255

F9259941C22BFCBC8C857CBBFBC0EE840F98703BF609B08C68E99C
605FC00D66D90A8F5F8D38D43C88
F7ABDBB28AC04694A0B867337F06D4F04F6F5AFBFAB8ECE7553
4D7F7D17780E1
2464AAF9599EFBCA6C54177437AB9EC8E073C6D',16)

sage: is_prime(n)
False

sage: from sage.libs.libecm import ecmfactor

sage: result = ecmfactor(n, 2e5, verbose=True)
Performing one curve with B1=200000
Found factor in step 1: 3684787

sage: result
(True, 3684787, 2673748448)

sage: n%result[1]
0
```

Le chiffrement de **socat** peut alors être grandement fragilisé, voire cassé. À l'heure où nous écrivons ces lignes, il n'est pas clair si cette vulnérabilité résulte d'une porte dérobée intentionnelle ou d'une erreur humaine.

2.2 Un test de primalité généraliste efficace

Les courbes elliptiques permettent aussi d'établir un test de primalité efficace et déterministe. En effet, la plupart des nombres premiers utilisés en cryptographie asymétrique sont testés à l'aide d'algorithmes de pseudo-primalité non déterministes (i.e. tests de Fermat et dérivés). On appelle alors ces nombres premiers des nombres premiers industriels. Même si la probabilité qu'un nombre premier industriel ne soit pas premier est très faible en pratique, il reste une chance qu'il le soit et qu'ainsi la clé générée soit faible.

Pour s'assurer qu'un candidat premier soit bien premier, il existe peu d'algorithmes adaptés. L'algorithme ECPP (cf. **[9]**), à base de courbes elliptiques, est un des rares utilisables pour les tailles de clés courantes. Il consiste à conditionner la primalité du candidat à un candidat plus petit, dont la primalité va elle-même être conditionnée par la primalité d'un candidat plus petit, et ainsi de suite jusqu'à ce que le dernier candidat soit suffisamment petit pour être testé de manière déterministe, par exemple par un test de petits diviseurs. Cet algorithme est par exemple implémenté dans l'outil Primo (cf. **[10]**).

2.3 Taille de clé

Un des principaux intérêts opérationnels des courbes elliptiques est la taille grandement réduite des clés utilisées par rapport aux autres cryptosystèmes asymétriques. Une taille de clé inférieure est un choix particulièrement adapté dans des périphériques embarqués ou des cartes à puces, où la mémoire et les capacités de calcul sont plus limitées. Les calculs s'en trouvent aussi plus rapides, pour un même niveau de sécurité attendu.

La raison de cette différence est simplement qu'il y a plus d'entiers utilisables comme clés que de nombres premiers, pour une taille de clé fixée, et que les courbes elliptiques reposent sur un secret entier et non premier. Par ailleurs, les meilleurs algorithmes connus pour résoudre le problème du logarithme discret sur les courbes elliptiques s'exécutent en un temps grosso modo proportionnel à la racine carrée de l'ordre de la courbe, le niveau de sécurité symétrique équivalent est donc la moitié de la taille de cet ordre.

Dans cet esprit, le NIST américain dresse une liste des tailles de clés ECC, RSA et AES sensiblement équivalentes :

Taille de clé ECC (bits)	Taille de clé RSA équivalente (bits)	Ratio de taille ECC/RSA	Taille de clé AES équivalente (bits)
163	1024	1:6	-
256	3072	1:12	128
384	7680	1:20	192
512	15360	1:30	256

Un autre intérêt connexe à cette taille de clé plus réduite est la difficulté à calculer le logarithme discret d'un point sur une courbe, qui est abordée dans la prochaine section.

3. ALGORITHMES

3.1 Logarithme discret

S'il est très facile de calculer n'importe quel multiple d'un point $Q = k.P$, l'opération inverse qui consiste, étant donnés P et Q, à retrouver l'index k, est appelée logarithme discret de Q en base Q, et est beaucoup plus complexe et consommatrice de temps. Cette opération (ECDLP en anglais) est à la base de la sécurité asymétrique apportée par les courbes elliptiques. Les meilleurs algorithmes connus sont de complexité proportionnelle à la racine carrée de l'ordre de la courbe (algorithme rho de Pollard et algorithme « baby-step giant-step ») :

Terminal

```
sage: p = random_prime(2^64)
sage: print "p =",p
p = 6334800468274178623
sage: E = EllipticCurve (GF(p), [1, 1])
sage: P = E.random_point()
sage: k = randint(0,p-1)
sage: print "k =",k
k = 4138954329113245806
sage: time Q = k*P
CPU times: user 4.07 ms, sys: 0 ns, total: 4.07 ms
Wall time: 3.49 ms
sage: time print P.discrete_log(Q)
971554093750979250
CPU times: user 32.2 s, sys: 119 ms, total: 32.4 s
Wall time: 32.3 s
```

Le dernier record du monde de calcul de logarithme discret sur courbe elliptique a été battu en janvier 2015, avec le calcul d'un index sur un corps premier de 113 bits, à l'aide d'une grappe de FPGA. (cf. **[11]**).

3.2 ECDH

ECDH (*Elliptic Curve Diffie-Hellman*) est l'exact équivalent de l'algorithme asymétrique classique de Diffie-Hellman, et permet l'échange d'une clé secrète entre deux protagonistes, sur un canal potentiellement écouté par des tiers. La clé n'est donc pas choisie par un des protagonistes, mais créée conjointement avec l'autre, à l'aide d'un échange qui peut être public. Alice choisit dans ce protocole un entier secret *a* et Bob un entier secret *b*. Toute l'astuce d'ECDH consistera pour Alice à calculer *a. (b. P)* et pour Bob, *b. (a. P)*. La multiplication d'un point par un entier étant commutative, Alice et Bob retrouveront le même point secret *K*.

Une implémentation très naïve en Sage pourrait être la suivante :

Terminal

```
def bob(E, G, A):
    p = E.base_field().order()
    b = randint(int(p/2),p-1)
    print "b =",b
    B = b*G
    print "-> B =",B
    K = b*A
    print "K(Bob) =",K
    return B

def alice():
    p = random_prime(2^128)
    E = EllipticCurve(GF(p), [1, 1])
    print "-> E =",E
    G = E.gens()[0]
    print "-> G =",G
    a = randint(int(p/2),p-1)
    print "a =",a
    A = a*G
    print "-> A =",A
    B = bob(E, G, A)
    K = a*B
    print "K(Alice) =",K

alice()
```

qui donne le résultat suivant :

Terminal

```
-> E = Elliptic Curve defined by y^2 = x^3 + x + 1 over Finite Field of size 3145561967301289085
1290000450919720778 3
-> G = (276751508922438582171275890129543896 4 : 503976656242852876341073044450096670 66 : 1)
a = 3109375876706508435335200922760726572 79
-> A = (198832356378906331772489667414828793 93 : 139688423001736486150909498639007725193 : 1)
b = 232866244852634667336145700063430844454
-> B = (293282367283288852592959372940278701780 : 147711437904342317974202339262556284455 : 1)
K(Bob) = (130500263234360331112278157617569640285 : 915426551349024962834708840619072239 5 : 1)
K(Alice) = (130500263234360331112278157617569640285 : 915426551349024962834708840619072239 5 : 1)
```

Alice et Bob ont donc pu échanger un point secret et l'ordonnée ou l'abscisse de ce point peut maintenant être utilisée comme clé secrète symétrique.

3.3 ECDSA

ECDSA (*Elliptic Curve Digital Signature Algorithm,* cf. **[12]**) est là encore l'équivalent sur courbes elliptiques de l'algorithme signature DSA, et nécessite comme ce dernier l'usage d'un nombre aléatoire à usage unique pour la signature. ECDSA repose très souvent sur des courbes recommandées par le NIST ou Certicom.

4. ATTAQUES CONNUES

4.1 Attaques contre ECDSA

Il est fondamental d'utiliser un nombre aléatoire *k* lors de toutes les signatures avec ECDSA. Si un même entier est réutilisé 2 fois par le signataire, il devient possible de casser complètement l'algorithme de signature et ainsi de signer des messages arbitraires (cf. **[13]**).

En décembre 2010, le groupe fail0verflow avait annoncé avoir cassé le système de signature électronique de la console PlayStation 3 de Sony en utilisant cette vulnérabilité, car dans le cas présent, Sony utilisait une valeur statique de cet entier. Plus récemment, plusieurs implémentations minoritaires de Bitcoin ont été touchées, soit parce que cet entier était constant, soit car son aléa était insuffisant et que certaines signatures finissaient par réutiliser le même entier.

4.2 Attaques contre le logarithme discret

La plupart des attaques contre le logarithme discret se font quand l'ordre de la courbe n'est pas premier. Il est donc crucial qu'il le soit ou qu'il soit multiple d'un très grand nombre premier, pour préserver la sécurité de la courbe.

Certaines courbes sont utilisées modulo p^m avec un petit exposant $m > 1$ pour des raisons de performance, mais récemment il a été montré que le logarithme discret dans ce cas était bien plus facile à calculer.

Enfin, dans le cas d'ECDH, il est important que le canal soit bien authentifié entre Alice et Bob, sinon une tierce partie peut substituer le point utilisé par un point ayant un ordre faible, puis par force brute énumérer les clés possibles après l'échange (« small subgroup attack » en anglais).

4.3 Portes dérobées

Il est fondamental de s'assurer que les courbes elliptiques que l'on utilise n'ont pas été manipulées au préalable, par exemple par introduction de constantes *a, b, p* ou d'un ordre particulier, qui ne seraient ni naturels, ni nécessaires au fonctionnement de la courbe, et qui pourraient résulter d'un choix volontaire pour affaiblir la sécurité de la courbe. C'est ce qui est arrivé récemment à l'algorithme de génération de nombres pseudo-aléatoires Dual EC DRBG, qui fait l'objet d'une étude approfondie dans *MISC n°84*.

À ce titre, Daniel J. Bernstein et Tanja Lange ont mis en place un site qui analyse la composition de courbes elliptiques classiques et tente d'en donner un niveau de sécurité (cf. figure 3, page suivante).

		Parameters:			ECDLP security:				ECC security:			
Curve	Safe?	field	equation	base	rho	transfer	disc	rigid	ladder	twist	complete	ind
Anomalous	False	True ✔	True ✔	True ✔	True ✔	False	False	True ✔	False	False	False	False
M-221	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔
E-222	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔
NIST P-224	False	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	False	False	False	False	False
Curve1174	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔
Curve25519	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔
BN(2,254)	False	True ✔	True ✔	True ✔	True ✔	False	False	True ✔	False	False	False	False
brainpoolP256t1	False	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	False	False	False	False
ANSSI FRP256v1	False	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	False	False	False	False	False
NIST P-256	False	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	False	False	True ✔	False	False
secp256k1	False	True ✔	True ✔	True ✔	True ✔	True ✔	False	True ✔	False	True ✔	False	False
E-382	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔
M-383	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔
Curve383187	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔
brainpoolP384t1	False	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	False	True ✔	False	False
NIST P-384	False	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	False	False	True ✔	False	False
Curve41417	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔
Ed448-Goldilocks	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔
M-511	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔
E-521	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔	True ✔

Fig. 3 : Comparaison des niveaux de sécurité de quelques courbes elliptiques d'après https://safecurves.cr.yp.to/.

4.4 Attaques par canaux auxiliaires

Lors de chaque phase du calcul de *k. P*, l'algorithme d'exponentiation binaire utilise soit un doublement de points, soit une addition du point P. Il se trouve que l'algorithme de doublement de point est plus rapide et moins consommateur de ressources que l'addition basique de deux points. Cette différence peut être perçue par une analyse continue de consommation de courant ou de rayonnement électromagnétique émis lors du calcul, pour ainsi retrouver par des mesures physiques la valeur de *k*, souvent utilisée comme clé secrète dans la cryptographie sur courbes elliptiques. Cette attaque a été réussie avec succès tout récemment contre ECDH et GnuPG Libgcrypt 1.6.3 (vulnérabilité CVE 2015-7511) par mesure du rayonnement électromagnétique émis par un ordinateur portable à travers un mur (cf. **[14]**). La contre-mesure recommandée est d'éviter toute branche conditionnelle dans ces étapes de calcul (par exemple à l'aide de l'échelle de Montgomery), ce qui dans le cas des courbes elliptiques, peut parfois se faire sans surcoût de temps de calcul.

4.5 Attaque par ordinateur quantique

Ces dernières années, l'informatique quantique a beaucoup progressé sur le plan matériel, et des systèmes à une quinzaine de qubits commencent à fonctionner en laboratoire. L'algorithme de Shor (cf. **[15]**), connu depuis une vingtaine d'années, permet de factoriser efficacement, en temps polynomial, une clé RSA publique, et ainsi de la casser rapidement. Une variante de cet algorithme, bien plus rapide, permet de façon similaire de retrouver le logarithme discret d'un point et ainsi de casser efficacement ECDH et ECDSA. Des organismes ou grosses sociétés influentes estiment qu'un ordinateur quantique fonctionnel existera dans les prochaines années : Google (entre 5 et 10 ans), le NIST, Microsoft et l'ETSI (dans 10 ans environ), la Commission Européenne (environ 15 ans).

Ainsi, le NIST vient de publier un rapport intitulé « Report on Post-Quantum Cryptography » (cf. **[16]**) et recommande de ne pas consacrer d'effort conséquent pour migrer les systèmes asymétriques classiques vers les courbes elliptiques, qu'il estime ayant des durées de vie trop courtes à l'heure actuelle. Dans la conclusion de son rapport, il recommande aux agences américaines d'abandonner toute cryptographie asymétrique classique d'ici seulement 10 ans. Ce même constat a été formulé en août 2015 par la NSA à travers les algorithmes de cryptographie asymétrique qu'elle utilise (appelés « suite B »), qu'elle souhaite voir évoluer. Les deux organismes travaillent avec de nombreux experts à des algorithmes résistants aux ordinateurs quantiques (cf. **[17]**). ■

RÉFÉRENCES ET LIENS

[1] Courbe elliptique secp256k1 : **https://en.bitcoin.it/wiki/Secp256k1**

[2] Logiciel de calcul open source Sage : **http://www.sagemath.org/fr/**

[3] Théorème de Hasse : **https://fr.wikipedia.org/wiki/Th%C3%A9or%C3%A8me_de_Hasse_sur_les_courbes_elliptiques**

[4] Algorithme de Schoof : **https://fr.wikipedia.org/wiki/Algorithme_de_Schoof**

[5] Théorème des restes chinois : **https://fr.wikipedia.org/wiki/Th%C3%A9or%C3%A8me_des_restes_chinois**

[6] Théorème de Lagrange : **https://fr.wikipedia.org/wiki/Th%C3%A9or%C3%A8me_de_Lagrange_sur_les_groupes**

[7] *MISC hors-série n°11*, « Outils de sécurité »

[8] Mise à jour vulnérable de socat : **http://repo.or.cz/socat.git/commitdiff/281d1bd6515c2f0f8984fc168fb3d3b91c20bdc0**

[9] Algorithme ECPP : **https://en.wikipedia.org/wiki/Elliptic_curve_primality**

[10] Logiciel Primo, implémentation de l'algorithme ECPP : **http://www.ellipsa.eu/public/primo/primo.html**

[11] Record du monde de calcul de logarithme discret sur courbes elliptiques : **http://eprint.iacr.org/2015/143**

[12] Algorithme ECDSA : **https://fr.wikipedia.org/wiki/Elliptic_curve_digital_signature_algorithm**

[13] Vulnérabilité courante sur ECDSA : **https://en.wikipedia.org/wiki/Digital_Signature_Algorithm#Sensitivity**

[14] Attaque par canaux cachés sur ECDH : **https://www.cs.tau.ac.il/~tromer/ecdh/**

[15] Algorithme de Shor : **https://fr.wikipedia.org/wiki/Algorithme_de_Shor**

[16] NIST, NISTIR 8105 - « Report on Post-Quantum Cryptography », février 2016 : **http://csrc.nist.gov/publications/drafts/nistir-8105/nistir_8105_draft.pdf**

[17] Cryptographie post-quantique : **https://en.wikipedia.org/wiki/Post-quantum_cryptography**

LE GRAND DÉFI DU POST-QUANTIQUE

Jean-Charles FAUGÈRE & Ludovic PERRET

En août 2015, la NSA a surpris le monde de la cybersécurité en faisant une annonce très surprenante à l'intention des entreprises et administrations américaines. Elle recommande de préparer le basculement de la cryptographie à clef publique classique fondée sur la théorie des nombres vers des systèmes résistants à l'ordinateur quantique. Depuis, l'organisme de normalisation américain NIST a lancé la course post-quantique avec un appel international pour la création de standards à l'épreuve de l'ordinateur quantique.

Le prix Turing – équivalent du prix Nobel en informatique – a récompensé cette année les deux cryptologues W. Diffie et M. Hellman pour leur protocole d'échange de clefs et l'invention de la cryptographie à clef publique. Ces idées révolutionnent la cryptographie à la fin des années 80 et permettent aujourd'hui à des millions d'utilisateurs du Web de communiquer de manière confidentielle. La sécurité du protocole de Diffie-Hellman, et plus généralement la cryptographie à clef publique, repose sur des problèmes mathématiques réputés difficiles. Par exemple, le protocole Diffie-Hellman est basé sur la difficulté de trouver un logarithme discret (DLOG) dans des corps finis ou des courbes elliptiques. Le chiffrement à clef publique RSA – du nom de ses inventeurs R. Rivest, A. Shamir, L. Adelman eux aussi prix Turing en 2002 – repose sur la difficulté de décomposer des grands nombres en produits de facteurs premiers (problème FACT).

La plupart des protocoles utilisant la cryptographie à clef publique (https, IPSEC...) reposent uniquement sur la difficulté des problèmes FACT et DLOG. Ainsi, une percée technologique ou mathématique remettant en cause la difficulté de ces problèmes rendrait vulnérables toutes les communications, et paralyserait, par exemple, le commerce électronique. C'est le scénario de la « cryptocalypse » ; une défaillance simultanée dans la difficulté des deux problèmes mathématiques qui garantissent la sécurité des échanges électroniques mondiaux.

En l'état de nos connaissances et avec la technologie actuelle, nous sommes encore loin de cette situation. Avec des paramètres appropriés, RSA et Diffie-Hellman sont encore considérés comme sûrs. Toutefois, on sait depuis 20 ans déjà que la cryptographie à clef publique basée sur DLOG et FACT est menacée par une percée technologique : les ordinateurs quantiques.

Il existe déjà des machines quantiques qui sont commercialisées par la start-up canadienne D-WAVE **[DWAVE]**. La puissance véritable de ces machines reste encore aujourd'hui un sujet controversé. D'autre part, la machine D-WAVE n'est pas un calculateur quantique complet et permet « uniquement » de résoudre des problèmes de type optimisation.

À notre connaissance, il n'existe pas aujourd'hui d'ordinateur quantique (complet) suffisamment puissant pour factoriser des grands nombres. Le plus grand nombre jamais factorisé avec un processus quantique est 56153. Pour attaquer RSA, il faut factoriser un nombre 130 fois plus gros.

La construction d'un ordinateur quantique reste un risque difficile à mesurer en cryptographie. Certains spécialistes du domaine annoncent l'arrivée imminente d'une telle machine. D'après M. Mariantoni **[M14]**, on pourrait construire une machine quantique d'ici 15 ans pour un budget de un milliard de dollars et il faudrait alimenter cette machine avec une centrale nucléaire. À l'échelle d'un État, cela semble plausible. Inversement, d'autres spécialistes pensent que l'apparition de la machine quantique sera plus lente : au moins 50 ans.

Par conséquent, le pays capable de construire un ordinateur quantique posséderait un atout stratégique majeur. La course technologique pour cette machine est lancée avec des investissements massifs à travers le monde: Union européenne **[EUq]**, Google avec D-WAVE et la NASA **[AiQ]**, Chine **[Ali,XXZH16]** et d'autres. C'est une version moderne de la course pour marcher sur la Lune.

Cependant, il est possible de construire des cryptosystèmes à clef publique avec d'autres problèmes mathématiques et le risque quantique est aujourd'hui jugé suffisamment critique pour que le NIST, organisme de standardisation américain, annonce début 2016 son intention de standardiser des algorithmes à clefs publiques résistants à l'ordinateur quantique **[NISTpq]**. Avec le recul, nous savons qu'un algorithme standardisé par le NIST devient de facto un standard mondial. En cryptographie, un exemple de standard bien connu du NIST est le chiffrement

à clef secrète AES. L'Europe n'est pas (trop) en retard sur le sujet puisque l'organisme de normalisation européen (ETSI) travaille sur la standardisation des algorithmes post-quantique **[ETSIpq]** depuis 2015.

Une mini-révolution en cryptographie à clef publique est en marche, et le défi s'annonce colossal **[Mo15]** : la transition de notre infrastructure à clef publique vers des algorithmes post-quantiques.

1. IMPACT DE L'ORDINATEUR QUANTIQUE EN CRYPTOGRAPHIE

L'ordinateur quantique est la promesse d'une machine qui utilise des phénomènes de physique quantique pour décupler sa puissance de calcul. L'ordinateur quantique permet ainsi de résoudre certains problèmes mathématiques beaucoup plus efficacement qu'une machine classique. S. Jordan, chercheur au NIST, tient à jour un « bestiaire » des algorithmes quantiques **[QZoo]** avec une liste quasiment exhaustive des problèmes pouvant se résoudre plus efficacement avec un ordinateur quantique.

L'exemple certainement le plus célèbre des capacités d'un ordinateur quantique est donné par P. Shor. Il propose un algorithme qui est capable de résoudre DLOG et FACT sur une machine quantique en un temps polynomial. Le choc est rude et la conséquence sans appel. Ainsi, dans le monde quantique, RSA et Diffie-Hellman offrent seulement le même niveau de sécurité que la méthode de chiffrement très simple utilisée par Jules César dans ses correspondances secrètes !

La cryptographie à clef secrète est également touchée par l'ordinateur quantique. L'algorithme de Grover permet une accélération quadratique de la recherche exhaustive. Prenons l'exemple de l'algorithme de chiffrement à clef symétrique AES avec une clef secrète de 128 bits. Sur une machine classique, la recherche exhaustive nécessite d'énumérer au plus 2128 clefs. Avec une machine quantique et l'algorithme de Grover, il faut parcourir au plus 264 candidats pour retrouver la bonne clef. L'impact est ici moins important puisqu'il suffit de doubler la taille des clefs en cryptographie symétrique pour se prémunir de l'ordinateur quantique.

Récemment M. Kaplan, G. Leurent, A. Leverrier, et M. Naya-Plasencia ont montré que l'impact de l'ordinateur quantique sur la cryptographie symétrique ne se limitait pas forcément à l'algorithme de Grover. Les auteurs de cet article utilisent un algorithme quantique encore jamais utilisé en cryptanalyse – l'algorithme de Simon – pour attaquer certains *modes opératoires* des chiffrements symétriques. Un chiffrement symétrique comme l'AES opère sur des blocs de taille fixe, 128 bits. Un mode opératoire est une technique permettant de faire opérer un chiffrement symétrique sur des messages de taille quelconque. L'attaque nécessite toutefois de faire une hypothèse assez forte sur les moyens de l'attaquant. Ceci indique que l'impact de l'ordinateur quantique en cryptographie est une histoire qui n'est pas encore complètement écrite et nous réserve des rebondissements.

Il est important de souligner qu'il existe des problèmes qui sont difficiles à résoudre indépendamment de la machine (quantique ou classique). En théorie de la complexité, nous classifions les problèmes en fonction de leur difficulté intrinsèque. Les problèmes NP-difficiles sont des problèmes pour lesquels il n'existe pas a priori d'algorithme efficace pour les résoudre ; en quantique comme en classique. C'est la fameuse conjecture P différent de NP.

2. CRYPTOGRAPHIE POST-QUANTIQUE

En pratique, nous utilisons la cryptographie à clef publique essentiellement pour l'échange des clefs (en utilisant éventuellement un algorithme de chiffrement à clef publique) et pour l'authentification par certificats (signature). L'objectif de la *cryptographie post-quantique* est de construire des cryptosystèmes (échange de clef, chiffrement, signature...) résistants aux ordinateurs quantiques. Cette cryptographie post-quantique inclue typiquement **[ETSIpq,NISTpq,BBD09]** *la cryptographie multivariée, la cryptographie fondée sur codes correcteurs d'erreurs, la cryptographie fondée sur réseaux euclidiens, et la cryptographie fondée sur des arbres de hachages.*

Il existe d'autres alternatives post-quantiques que nous ne traiterons pas ici comme la cryptographie à base d'isogénies, celle-ci étant plus récente. Nous trouvons également des cryptosystèmes qui utilisent directement la physique quantique. Typiquement, il est possible de construire un protocole d'échange de clef dont la sécurité repose sur des lois physiques. La distribution quantique des clefs est aujourd'hui commercialisée, mais le coût reste trop élevé pour un déploiement à grande échelle.

2.1 Cryptographie multivariée

La cryptographie multivariée consiste à construire des cryptosystèmes dont la sécurité repose sur la difficulté du problème PoSSo : c'est-à-dire de trouver - s'il existe - un zéro commun d'un ensemble de polynômes non-linéaires. Le problème PoSSo est NP-difficile et sa difficulté n'est a priori pas remise en cause par l'émergence d'un ordinateur quantique.

Les bases de Gröbner sont un outil important pour évaluer la sécurité des cryptosystèmes multivariés. Ce concept peut également servir à analyser la sécurité d'autres primitives post-quantiques. Les bases de Gröbner permettent, notamment, de trouver les solutions d'un système d'équations non-linéaires. La complexité des meilleurs algorithmes de base de Gröbner sert souvent de référence pour spécifier en pratique les paramètres des cryptosystèmes multivariés.

En général, la clef publique d'un cryptosystème multivarié est donnée par un ensemble de polynômes non-linéaires. Pour chiffrer un message, il suffit d'évaluer le message sur les polynômes de la clef publique. On donne ci-dessous un exemple jouet de chiffrement en cryptographie multivariée. Nous avons utilisé un logiciel de calcul forme MAGMA pour réaliser cet exemple:

Fichier

```
/* Le message à chiffrer est donné sous forme d'un vecteur de bits */
 [0, 1, 1, 1, 1];

/* Petit exemple d'une clef publique; 5 polynômes en 5 variables.
 [ x1*x2 + x1 + x2*x3 + x2*x5 + x2 + x3*x5 + x3 + x4 + 1,
   x1*x2 + x1*x3 + x1*x4 + x1 + x2*x3 + x2*x4 + x3 + x4*x5,
   x1*x3 + x1*x4 + x1*x5 + x1 + x3 + x4*x5,
   x1*x2 + x2*x5 + x2 + x3*x4 + x3*x5 + x3 + x4*x5 + x4 + x5 + 1,
   x1*x3 + x1*x5 + x1 + x2*x3 + x2*x4 + x2 + x3 + x4 ]

/* Pour chiffrer, on évalue les polynômes de la clef publique pour le message
x1=0, x2=1, x3=1, x4=1, x5=1, et obtient le message chiffré en prenant le
reste modulo 2: */

 [ 1, 0, 0, 1, 1]
```

La cryptographie multivariée permet aussi d'obtenir des schémas de signature. La clef publique sera toujours donnée par un ensemble de polynômes non-linéaires. La vérification d'une signature consiste simplement à évaluer les polynômes de la clef publique sur la signature.

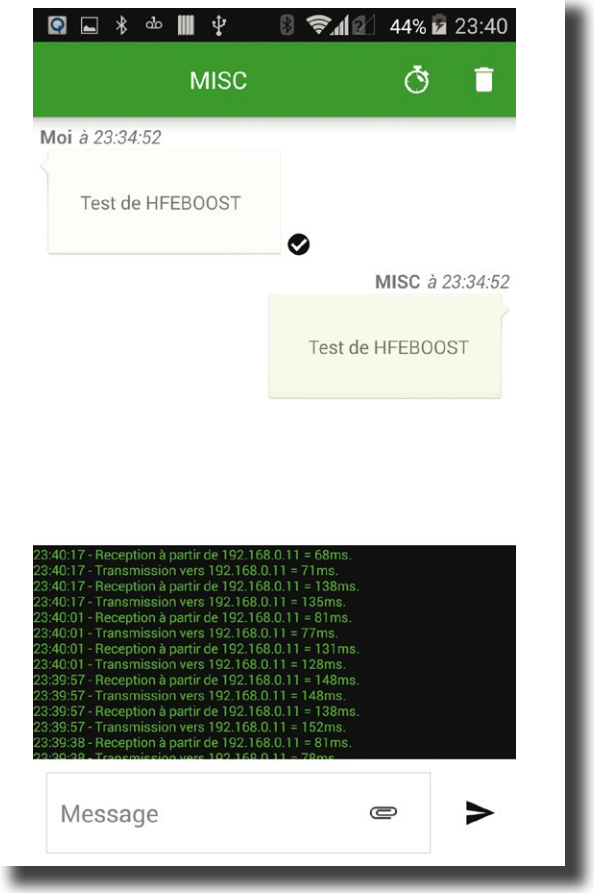
Le domaine est très dynamique, et de nombreuses constructions sont présentées chaque année par différents auteurs à travers le monde. Certains ne résistent pas à l'analyse de la communauté. Parmi les schémas multivariés ayant résisté à l'analyse des cryptologues, on compte HFE- (*Hidden Field Equations*) **[HFE96]**, UOV (*Unbalanced Oil and Vinegar*) et QUARTZ. Ces schémas suivent tous le principe de chiffrement (ou de vérification d'une signature) que nous venons d'évoquer. Ils diffèrent ensuite sur la méthode de construction de la clef secrète, et donc de la trappe.

Dans HFE, la clef publique est un ensemble de n polynômes non-linéaires en n variables dont les coefficients sont sur F_2 le corps à deux éléments. L'idée est de construire la clef publique à partir d'un polynôme P particulier en une variable définie sur F_{2^n} ; une extension de degré n de F_2.

Fichier

```
/* Exemple de polynôme de type HFE sur une extension de degré 5; w est un
générateur du corps F_2^5 et X est la variable  */

  w^7*X^17 + w^29*X^12 + w^11*X^10 + w^29*X^9 + w^26*X^6 + w^28*X^5 +
w^6*X^3
```



Dans HFEBoost **[PolSys]**, une autre variante du schéma HFE-, que nous avons développé pour des tests de l'armée de terre, la clef publique était de 130 Ko. Typiquement, une clef publique RSA est de 2000 bits. Dans des infrastructures réseaux modernes, ce point n'est plus vraiment handicapant. Les tests terrains pour HFEBoost ont été réalisés sur un réseau 4G. Les clefs publiques transitaient entre des centaines de participants sans problème de latence.

Les polynômes de la clef publique sont une *version masquée* des composantes du polynôme P *déplié* sur F_2. Avec la clef secrète, le déchiffrement est alors équivalent à trouver les racines du polynôme en une variable P. Contrairement au problème de trouver les zéros d'un ensemble polynômes non-linéaires qui problème NP-difficile, le problème de trouver les racines est lui bien plus facile. On trouve les racines d'un polynôme en une variable très efficacement ; en un temps quasi-linéaire en son degré.

Le point fort de la cryptographie multivariée est de construire des schémas permettant de signer des messages avec des signatures très courtes. Par exemple, l'algorithme QUARTZ, une variante de HFE-, permet d'obtenir des signatures de l'ordre de 100 bits.

Un point longtemps bloquant était la taille des clefs publiques ; bien plus élevée que les cryptosystèmes classiques basés sur DLOG ou FACT. C'est une caractéristique commune à presque tous les cryptosystèmes post-quantiques (Figure ci-contre).

2.2 Cryptographie fondée sur les codes correcteurs

Le cryptosystème de McEliece est le chiffrement à clef publique post-quantique le plus ancien. Sa conception date de 1978 ; juste après l'invention de la clef publique par W. Diffie et M. Hellman. La sécurité du cryptosystème de McEliece repose sur la difficulté de décoder un code linéaire. Nous pouvons voir ce problème comme celui de trouver la solution d'un système linéaire dont une partie des équations est erronée. Il est très simple de résoudre un système d'équations linéaires, mais la tâche est bien plus complexe si les équations comportent des erreurs. Ce problème, dénommé BoundedDecoding, est notoirement difficile. Il a été prouvé NP-Dur et largement étudié. La complexité du meilleur algorithme pour résoudre BoundedDecoding sert de référence pour choisir les paramètres de McEliece.

La clef publique est ici donnée par une matrice à coefficients binaires. Cette matrice n'est pas aléatoire, mais est dérivée d'un *code correcteur d'erreur* particulier: *un code de Goppa binaire*. En résumé, un code correcteur d'erreurs (linéaire) est un espace vectoriel. Nous pouvons ainsi représenter un code linéaire par une matrice génératrice dont les lignes sont les vecteurs d'une base de cet espace vectoriel. Pour les codes de Goppa binaires, nous avons des méthodes efficaces permettant de résoudre le problème BoundedDecoding.

Dans McEliece, le message que l'on souhaite chiffrer est représenté sous la forme d'un vecteur. Pour chiffrer, il faut d'abord choisir un *vecteur d'erreurs*. On chiffre ensuite en multipliant notre message par la matrice publique, puis on ajoute au résultat le vecteur d'erreurs. Pour illustrer le principe, on donne ci-dessous un exemple jouet de chiffrement avec McEliece :

Fichier

```
/* Le message à chiffrer est donné par un vecteur 5 bits */
 [0, 1, 1, 1, 1];

/*  Exemple d'une clef publique; matrice M à coefficients binaires  de 5
lignes et 10 colonnes */
 [1 0 0 0 0 0 1 1 1 1]
 [0 1 0 0 0 1 0 1 0 0]
        [0 0 1 0 0 1 0 1 1 1]
 [0 0 0 1 0 0 0 0 1 1]
 [0 0 0 0 1 1 0 1 1 1]
```

```
/* Pour chiffrer, il faut choisir un vecteur d'erreurs */
 [ 1, 0, 0, 0, 0, 0, 0, 1, 0, 1 ]

/* Ensuite, il faut multiplier le message par la matrice publique M, et
ensuite additionner le résultat avec le vecteur d'erreurs (les calculs se
font modulo 2). */
/* Voici le message chiffré. */
 [1 1 1 1 1 1 0 1 0 1]
```

Il est également possible d'obtenir un schéma de signature en utilisant l'idée de McEliece [CFS01]. Ce schéma est toutefois moins compétitif qu'une signature multivariée.

Le système de chiffrement proposé par McEliece a résisté à toutes les tentatives de cryptanalyse depuis 1978. C'est remarquable au regard de l'intense activité en cryptographie. En plus de sa résistance à l'ordinateur quantique, le schéma de McEliece présente plusieurs autres avantages vis-à-vis de la cryptographie classique, comme sa vitesse de chiffrement et de déchiffrement. Comme en cryptographie multivariée, un point bloquant pour McEliece était la taille des clés publiques, de l'ordre de 100 fois plus élevé qu'une clef RSA typique. Des tentatives ont été faites pour réduire ces tailles de clé en utilisant des codes plus structurés. Plusieurs travaux, dont **[FOPPT15]**, ont montré que cette approche donnait lieu à des attaques par bases de Gröbner qui se révèlent très dangereuses sur ces familles de codes structurés. La dernière tentative en date pour réduire la taille des clefs consiste à utiliser des codes MDPC.

En définitive, le schéma de chiffrement McEliece dans sa version avec des Goppa binaires reste une référence de sécurité pour cette cryptographie.

2.3 Cryptographie reposant sur les réseaux euclidiens

Cette cryptographie repose sur un autre objet mathématique bien étudié : *les réseaux euclidiens*. Pendant des années, les réseaux euclidiens étaient surtout réputés comme un outil de cryptanalyse. Leur utilisation cryptographique a connu un premier renouveau grâce aux travaux de J. Hoffstein, J. Pipher, et J. H. Silverman sur NTRU et une explosion avec les résultats de O. Regev **[Rev05]**.

O. Regev propose un schéma de chiffrement à clef publique similaire au chiffrement de McEliece. La clef publique est (essentiellement) donnée par une matrice *aléatoire* dont les coefficients sont des entiers modulo un nombre premier. Il s'agit d'une différence importante avec McEliece, puisque la matrice de la clef publique n'est pas aléatoire.

Fichier

```
>  /* Exemple de la  matrice publique dans le schéma de O. Regev. */
 [ 10  18  17 110 103 110  96 115  46  76]
 [108  58  15 117  43  29   3  55  17  37]
 [ 74  66  44  82  34  97  63  16  17  11]
 [ 21 107  91 108  43  49  27 112  45  20]
 [121  48  41 122  75  59  91  15  93  69]
```

Dans le schéma de Regev, le chiffrement consiste également à multiplier un vecteur par la matrice publique et tirer un vecteur d'erreurs. Les coefficients du vecteur d'erreurs sont choisis selon une loi de distribution particulière : *une loi Gaussienne discrète*. C'est une autre différence avec McEliece.

Le chiffrement de Regev est particulièrement séduisant, car il est possible de relier la sécurité du cryptosystème à la difficulté de résoudre l'instance la plus difficile d'un problème portant sur les réseaux euclidiens. Cette propriété permet de garantir un haut niveau de confiance.

Les réseaux *euclidiens* offrent aujourd'hui une grande flexibilité pour la conception de cryptosystèmes.

Nous trouvons des protocoles de chiffrement, de signature (BLISS) et bien d'autres ; dont la sécurité repose sur des problèmes difficiles liés aux réseaux *euclidiens*.

Un défi pour cette cryptographie est de réduire l'écart qui existe entre les constructions qui sont « prouvées sûres » et leurs variantes utilisables en pratique. En effet, la taille de certains paramètres nécessaire pour garantir une sécurité prouvée est trop élevée en pratique. Il est donc fréquent d'utiliser les meilleures attaques connues pour réduire la taille de ces paramètres en conservant une certaine sécurité.

2.4 Cryptographie fondée sur les arbres de hachages

Le principe de la cryptographie fondée sur les arbres de hachages remonte à L. Lamport en 1979. L'idée est d'utiliser uniquement une fonction de hachage cryptographique. Cette technique permet de construire uniquement des schémas de signature dont la sécurité repose sur la fonction de hachage utilisée. Le schéma proposé par L. Lamport a toutefois un intérêt limité puisque nous pouvons seulement signer un seul message avec la même clef publique sans compromettre la sécurité. R. Merkel propose en 1989 une amélioration du schéma de L. Lamport permettant de signer un nombre plus important de messages en utilisant un arbre de hachage (ou arbre de Merkle). Le nombre de signatures que l'on autorise pour une même clef publique dépend de la profondeur de l'arbre et influence aussi les performances du schéma (taille de la signature). Dans **[XMSS,SPHINCS]**, les auteurs ont amélioré le principe de Merkle. Finalement, nous pouvons obtenir – pour cette famille post-quantique – des schémas de signature avec des performances acceptables. Typiquement, la clef publique dans SPHINCS est de 1Kb, la signature de 41 Kb et permet de signer 250 messages sans compromettre la sécurité.

3. LA COURSE AU POST-QUANTIQUE

Pour un spécialiste du domaine, le changement soudain du statut de la cryptographie post-quantique est impressionnant. En 2012, la conférence « Symbolic Computation and Cryptography », dédiée à l'analyse des systèmes post-quantiques, réunissait une quarantaine de chercheurs. L'année suivante, la conférence « Post-Quantum Cryptography (PQCrypto) », organisée à Limoges, réunissait une audience légèrement supérieure.

En 2016, c'est le boom du domaine. La conférence PQCrypto, qui se déroule au Japon, enregistre un record d'affluence : plus de 222 inscrits. Dans cette foule, nous trouvons des chercheurs, mais également de nombreux industriels (Cisco, Google, Gemalto, Intel, LG, Microsoft, NTT, Toshiba...) qui se pressent pour comprendre le phénomène et prendre le train post-quantique en route.

L'événement déclencheur est une annonce inattendue de la *National Security Agency* (NSA), en août 2015, qui conseille aux administrations américaines d'anticiper dès à présent le basculement vers une cryptographie post-quantique **[NSASuiteb]**. Cette annonce arrive dans une période où la cryptographie à base de courbes elliptiques devait envahir notre quotidien et enfin se substituer à l'utilisation de RSA. D'après la NSA, les courbes elliptiques ne sont pas une technologie d'avenir :

« Unfortunately, the growth of elliptic curve use has bumped up against the fact of continued progress in the research on quantum computing, which has made it clear that elliptic curve cryptography is not the long term solution many once hoped it would be. »

Au regard de la réputation de cette agence et des révélations de l'affaire Snowden, l'annonce ne manque pas de faire réagir : la NSA possède-t-elle un ordinateur quantique ? La NSA manipule-t-elle encore une fois les standards à son avantage ? La NSA sait-elle « casser » les courbes elliptiques ? La NSA pense-t-elle que d'autres pays sont en passe de construire un ordinateur quantique ? ... Nous laissons le lecteur se forger une opinion sur la question. Une référence sur ce sujet est l'article de N. Koblitz et A. Menezes **[KM15]**.

Aujourd'hui, le risque de l'ordinateur quantique est perçu comme très élevé. Concernant les secrets ayant une longue durée de vie (typiquement les données gouvernementales et les données médicales) la menace est jugée crédible. Le PIDS, un équivalent hollandais de l'Agence Nationale de la Sécurité des Systèmes d'Information (ANSSI), recommande dès aujourd'hui de « surchiffrer » ces données avec un chiffrement classique et un chiffrement post-quantique. D'autre part, le déploiement total d'un nouveau standard cryptographique nécessite au moins 20 ans. Le lecteur de *MISC* sait que ce déploiement est un parcours semé d'embûches. La solidité mathématique d'un algorithme n'est pas l'unique composante dans la chaîne de la sécurité. Il faut aussi prendre en compte les problèmes au niveau de l'implémentation et l'interaction de la brique post-quantique avec des protocoles de plus haut niveau comme TLS, SSH et IPSEC. Ces derniers aspects ont été très peu étudiés jusqu'à présent. Le NIST, l'ETSI et d'autres institutions, jugent ainsi qu'il est nécessaire d'agir dès maintenant.

Le NIST a ainsi lancé un appel international pour normaliser des algorithmes post-quantiques **[M16]**. En priorité, le NIST souhaite avoir des standards pour deux fonctionnalités : signature numérique et échange des clefs. Le NIST a appelé la communauté mondiale à soumettre ses meilleurs algorithmes post-quantiques d'ici la fin 2017. Il s'ensuivra une période d'étude des algorithmes d'environ trois ans au bout de laquelle le NIST sélectionnera un ou plusieurs algorithmes post-quantiques en fonction du niveau de confiance que la communauté scientifique accorde aux candidats. Ce n'est pas une compétition puisque le NIST s'autorise à sélectionner plusieurs vainqueurs. Les modalités de l'appel sont encore en discussion et seront fixées fin 2016.

L'appel du NIST est à la fois un risque et une opportunité pour la filière française de cybersécurité. Il faut prendre conscience de cette mini-révolution pour éviter une perte de compétitivité à moyen terme. De l'autre côté de l'Atlantique, des entreprises comme Intel et Microsoft aiguisent déjà leur stratégie post-quantique

[Intelpq,Micrpq]. Dans cette course, nous avons la chance de posséder en France un atout important. Une proportion substantielle des compétences mondiales sur le post-quantique se trouve dans les équipes académiques françaises (Paris, Lyon, Rennes, Limoges, Marseille, Saint-Étienne, Toulon...). Nos chercheurs sont donc en capacité d'avoir un impact important sur les standards post-quantiques. C'est une belle opportunité pour les industriels d'accompagner et d'appuyer l'effort des académiques dans cette course au post-quantique. ■

REMERCIEMENTS

Nous remercions l'équipe de *MISC*, et particulièrement Emilien Gaspar pour ses commentaires et sa relecture attentive de l'article.

RÉFÉRENCES

⇨ **[Ali]** Alibaba. « Alibaba's Cloud Unit Teams with Chinese Researchers on Quantum Computing » :
http://fortune.com/2015/07/30/alibaba-chinese-academy-team-on-quantum-computing/

⇨ **[BBD09]** D. J. Bernstein, J. Buchmann et E. Dahmen, editors. « Post-Quantum Cryptography », Mathematics and Statistics Springer-11649. 2009

⇨ **[PolSys]** **http://www-polsys.lip6.fr/Links/index.html**

⇨ **[SPHINCS]** D. J. Bernstein, D. Hopwood, A. Hülsing, T. Lange, R. Niederhagen, L. Papachristodoulou, M. Schneider, P. Schwabe, et Z. Wilcox-O'Hearn. « SPHINCS : Practical Stateless Hash-Based Signatures », EUROCRYPT, LNCS 9056, pages 368-397, Springer, 2015

⇨ **[Intelpq]** E. Brickell, « The Intel Strategy for Post Quantum Cryptography », Invited talk, PQCrypto 2014

⇨ **[XMSS]** J. Buchmann, E. Dahmen, et A. Hülsing. « XMSS - a Practical Forward Secure Signature Scheme Based on Minimal Security Assumptions », PQ Crypto, LNCS 7071, pages 117–129. Springer, 2011

⇨ **[NISTpq]** L. Chen, S. Jordan, Y-K. Liu, D. Moody, R. Peralta, R. Perlner, et D. Smith-Tone. « Report on Post-Quantum Cryptography », NISTIR 8105 DRAFT, 2016

⇨ **[CFS01]** N. Courtois, M. Finiasz, et N. Sendrier. « How to Achieve a McEliece-Based Digital Signature Scheme », ASIACRYPT, LNCS 2248, pages 157-174, Springer, 2001

⇨ **[DWAVE]** D-WAVE. «The Quantum Computing Compagny » :
http://www.dwavesys.com/

⇨ **[ETSIpq]** ETSI, « ETSI Launches Quantum Safe Cryptography Specification Group » : **http://www.etsi.org/news-events/news/947-2015-03-news-etsi-launches-quantum-safe-cryptography-specification-group**, 2015

⇨ **[FOPPT15]** J.-C. Faugère, A. Otmani, L. Perret, F. de Portzamparc et J-P. Tillich. « Structural Cryptanalysis of McEliece Schemes with Compact Keys », Des. Codes Cryptogr., 2015

⇨ **[AIQ]** Google. « Launching the Quantum Artificial Intelligence Lab » : **http://googleresearch.blogspot.fr/2013/05/launching-quantum-artificial.html**

⇨ **[Qzoo]** S. Jordan. « The Quantum Algorithm Zoo » : **http://math.nist.gov/quantum/zoo/**

⇨ **[KM15]** N. Koblitz and A. Menezes, « A Riddle Wrapped in an Enigma », Cryptology ePrint Archive : Report 2015/1018 : **https://eprint.iacr.org/2015/1018.pdf**

⇨ **[Micrpq]** Brian LaMacchia : **https://news.microsoft.com/features/from-ai-and-data-science-to-cryptography-microsoft-researchers-offer-16-predictions-for-16/**

⇨ **[M14]** M. Mariantoni. « Building a Superconducting Quantum Computer », Invited talk, PQ Crypto 2014

⇨ **[M16]** D. Moody. « Post-Quantum Cryptography : NIST's Plan for the Future », PQCrypto 2016 : **https://pqcrypto2016.jp/data/pqc2016_nist_announcement.pdf**

⇨ **[Mo15]** M. Mosca. « Cybersecurity in an Era with Quantum Computers : Will we be Ready ? », IACR Cryptology ePrint Archive 2015, 1075, 2015

⇨ **[NSASuiteb]** NSA. « Cryptography Today » : **https://www.nsa.gov/ia/programs/suiteb_cryptography/** et **https://www.iad.gov/iad/library/ia-guidance/ia-solutions-for-classified/algorithm-guidance/cnsa-suite-and-quantum-computing-faq.cfm**

⇨ **[HFE]** J. Patarin. « Hidden fields Equations (HFE) and Isomorphisms of Polynomials (IP) : Two New families of Asymmetric Algorithms », EUROCRYPT, LNCS 1070, pages 33–48. Springer, 1996

⇨ **[Reg05]** O. Regev. « On Lattices, Learning with Errors, Random Linear Codes, and Cryptography », STOC, pages 84-93. ACM, 2005

⇨ **[EUQ]** Union Européenne, « Call to accelerate Quantum Technologies across Europe » : **https://ec.europa.eu/digital-single-market/en/news/call-accelerate-quantum-technologies-across-europe**

⇨ **[XXZH16]** H. Xiang, T. Xiang, Z.-F. Zhang et Z.-F. Han. « An Overview of PQC Workshops/Projects and Standardization Concerns in China » : **https://pqcrypto2016.jp/data/10-An_Overview_of_PQC_in_China_by_Hong_Xiang.pdf**

VISITEZ NOTRE BOUTIQUE ET DÉCOUVREZ NOS GUIDES !

LINUX MAGAZINE / FRANCE | GNU LINUX PRATIQUE ESSENTIEL | MISC Multi-System & Internet Security Cookbook | HACKABLE MAGAZINE | Open Silicium

ET VOUS ?
COMMENT LISEZ-VOUS VOS MAGAZINES PRÉFÉRÉS ?

« Moi, je les lis en version PAPIER ! »

« Moi, je les lis en version PDF ! »

« Moi, je consulte la BASE DOCUMENTAIRE ! »

RENDEZ-VOUS SUR **www.ed-diamond.com** POUR DÉCOUVRIR TOUTES LES MANIÈRES DE LIRE VOS MAGAZINES PRÉFÉRÉS !

LE CHIFFREMENT POUR TOUS

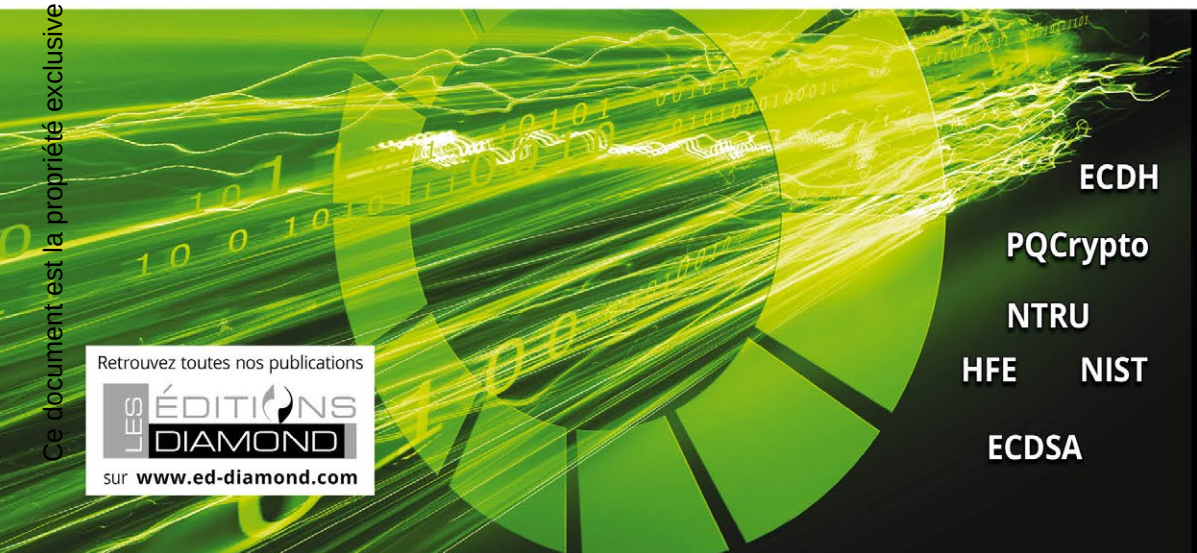
Découvrez les protocoles et les projets qui veulent vous protéger

LES TIERS DE CONFIANCE EN QUESTION

Les dernières évolutions qui bouleversent le monde des autorités de certification

L'ENJEU DE LA NORMALISATION

Découvrez les derniers standards qui occuperont une place importante dans l'écosystème de demain



LES AVANCÉES D'AUJOURD'HUI ET DE DEMAIN

Initiez-vous aux courbes elliptiques et à l'informatique quantique