Notes de cours
Semaine 12

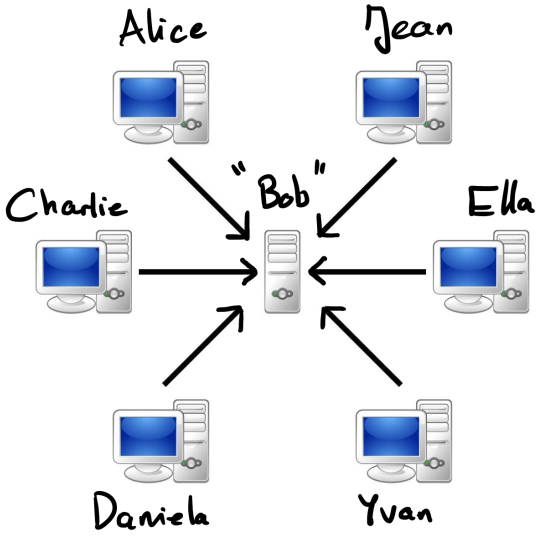
Cours Turing

1 Cryptographie à clé publique : suite

Dans ce chapitre, nous allons voir un nouveau type de cryptographie à clé publique, où le but n'est pas de partager une clé commune pour ensuite échanger des messages chiffrés, mais de directement communiquer un message chiffré ! Nous allons voir comment cette magie est possible avec un des protocoles les plus célèbres de cryptographie à clé publique : le *protocole de Rivest-Shamir-Adleman* (RSA). Puis nous verrons comment on peut avec le même principe certifier un document avec une *signature digitale* (DSA).

1.1 Protocole de Rivest-Shamir-Adleman (RSA)

Le scénario auquel s'applique ce protocole est un peu différent de celui où deux personnes essayent de communiquer secrètement des messages entre elles. Il faut plutôt penser à Alice comme une utilisatrice et à Bob comme un serveur central, auquel s'adressent de nombreuses personnes (typiquement, un site web auquel Alice désire se connecter). La contrainte maintenant est d'établir une(des) connection(s) sécurisée(s) sans qu'il soit nécessaire pour Bob de faire autre chose que de publier une clé qu'Alice et d'autres personnes puissent utiliser pour chiffrer directement leurs messages :



Encore un peu d'arithmétique modulaire !

Dans ce chapitre, nous aurons besoin d'un nouvel élément d'arithmétique modulaire. Rappelez-vous le petit théorème de Fermat :

$$\text{Si } N \text{ est un nombre premier et } 1 \leq A \leq N-1, \text{ alors } A^{N-1} \pmod N = 1.$$

Il existe une version plus générale, le *thèorème d'Euler*[1], donc l'énoncé est le suivant :

$$\text{Si } N \text{ est un nombre entier et } 1 \leq A \leq N-1 \text{ avec } \mathrm{PGDC}(A,N) = 1, \text{ alors } A^{\phi(N)} \pmod N = 1,$$

où ϕ est la *fonction d'Euler*. Cette fonction est définie ainsi :

$$\phi(N) \text{ est le nombre de nombres entiers } 1 \leq A \leq N-1 \text{ tels que } \mathrm{PGDC}(A,N) = 1.$$

En particulier, on trouve que :

- si N est premier, alors $\phi(N) = N-1$, car dans ce cas, *tous* les nombres A compris entre 1 et $N-1$ vérifient $\mathrm{PGDC}(A,N) = 1$: on retrouve ici le petit théorème de Fermat.

- si $N = P \cdot Q$, avec P et Q premiers, vous pouvez vérifier que $\phi(N) = (P-1)\cdot(Q-1)$. Par exemple, si $N = 10 = 2 \cdot 5$, alors $\phi(N) = 1 \cdot 4 = 4$; en effet, les nombres A compris entre 1 et 9 tels que $\mathrm{PGDC}(A,10) = 1$ sont $A = 1, 3, 7, 9$.

Remarque importante : Il est difficile de calculer la valeur de $\phi(N)$ si on ne connaît pas la décomposition de N en produit de facteurs premiers. Si par contre on la connaît, alors le calcul devient facile (il suffit d'effectuer le produit $(P-1)\cdot(Q-1)$ dans l'exemple précédent).

Revenons maintenant au protocole RSA. Voici comment celui-ci fonctionne :

1. Bob génère tout d'abord deux grands nombres premiers P et Q (distincts), et calcule $N = P \cdot Q$. Il publie ce nombre N (mais garde les nombres premiers P et Q secrets).
2. Il choisit ensuite un nombre C entre 2 et $N-1$ tel que $\mathrm{PGDC}(C, \phi(N)) = 1$. Bob publie également ce nombre C.
3. Il calcule alors le nombre D tel que $C \cdot D = 1 \pmod{\phi(N)}$: ce nombre D est donc l'inverse de C modulo $\phi(N)$. On peut montrer que cette opération est facile à effectuer, si on connaît la valeur de $\phi(N)$, bien sûr. Bob garde ce nombre D secret.
4. Puis vient l'étape du *chiffrement* : pour envoyer un message X compris entre 2 et $N-1$ à Bob, Alice calcule d'abord $Y = X^C \pmod N$ et envoie Y.
5. Et finalement vient l'étape du *déchiffrement* : Bob, pour retrouver le message envoyé par Alice, effectue l'opération $Y^D \pmod N$ et retrouve ainsi X, comme nous allons le voir.

1. A noter qu'il existe par ailleurs une démonstration courte et très élégante de ce théorème si on connaît des choses sur les groupes. Voir `https://en.wikipedia.org/wiki/Euler's_theorem`

Vérifions que tout ceci fonctionne bien :

1. Quel est le message reçu par Bob ? Calculons :

$$Y^D \pmod N = (X^C \pmod N)^D \pmod N = X^{C \cdot D} \pmod N = X \,?$$

Par construction, $C \cdot D = 1 \pmod{\phi(N)}$; ceci veut dire que $C \cdot D = 1 + K \cdot \phi(N)$ pour un nombre entier K et donc

$$Y^D \pmod N = X^{1+K\cdot\phi(N)} \pmod N$$

- Si $\text{PGDC}(X, N) = 1$, alors c'est aussi vrai que $\text{PGDC}(X^K, N) = 1$, et donc

$$X^{K\cdot\phi(N)} \pmod N = (X^K)^{\phi(N)} \pmod N = 1$$

par le théorème d'Euler, ce qui implique que $Y^D \pmod N = X$, car $2 \leq X \leq N - 1$.

- Si $\text{PGDC}(X, N) \neq 1$, alors une autre démonstration est nécessaire, mais notez que ce cas de figure arrive seulement dans le cas où X est un multiple de P ou de Q, donc très rarement.

2. Et que peut faire maintenant Eve, qui connaît les valeurs de N, C et Y ? Pour retrouver X, elle devrait d'abord retrouver la valeur du nombre D (afin d'effectuer ensuite le même calcul que Bob). Mais pour cela, elle a besoin de connaître le produit $\phi(N) = (P - 1)(Q - 1)$, or elle ne connaît que le nombre $N = P \cdot Q$. Et c'est là que vient s'ajouter la dernière pièce du puzzle, déjà mentionnée ci-dessus : retrouver les nombres P et Q à partir de N revient à *factoriser* ce nombre. Or il se trouve que cette opération est a priori également une opération difficile à réaliser (du moins, personne n'a trouvé jusqu'à maintenant d'algorithme efficace pour résoudre ce problème).

A propos de factorisation...

Pour finir, mentionnons que la sécurité du protocole RSA est compromise à l'heure actuelle par le développement des *ordinateurs quantiques*, qui pourraient bien un jour permettre d'effectuer efficacement des factorisations de grands nombres entiers, grâce à l'*algorithme de Shor*[2]. Lorsque les premiers ordinateurs quantiques ont vu le jour au début des années 2'000, ceux-ci pouvaient essentiellement effectuer des factorisations du type $15 = 3 \cdot 5$, et tout le monde a rigolé... Mais il faut bien avouer qu'aujourd'hui, soit vingt ans plus tard, plus personne ne rigole, car même s'il subsiste des incertitudes, il se pourrait bien que les prochaines années voient fleurir des choses très intéressantes de ce côté-là ! A tel point que le sujet de la *cryptographie post-quantique*, celle qui résisterait aux ordinateurs quantiques, est maintenant tout à fait sérieusement étudiée dans le monde académique[3].

2. Un sujet passionnant, mais qui mériterait rien qu'à lui un cours complet...

3. Signalons ici qu'il existe aussi une *cryptographie quantique*, qui utilise les propriétés intrinsèques des particules de lumière (les photons) pour chiffrer des messages.

1.2 Signature digitale (DSA)

Le but de cette seconde partie est différent de la première, mais utilise le même principe. L'idée est de proposer à Alice un moyen digital pour *signer* un message, afin que Bob puisse être sûr que le message en question provient bien d'Alice et pas de quelqu'un d'autre (Eve, par exemple). Faites attention qu'on oublie donc ici l'idée de transmettre *secrètement* un message !

Avant de présenter le protocole de signature digitale, nous avons besoin d'introduire un nouvel élément : les *fonctions de hachage*.

Fonctions de hachage

A l'origine, une fonction de hachage H a pour but de faire correspondre à un message donné X une *valeur de hachage* $H(X)$ qui satisfasse les propriétés suivantes :

1. $H(X)$ doit être déterministe (i.e., non aléatoire) ;
2. $H(X)$ doit occuper (significativement) moins de place en mémoire que X lui-même ;
3. si X et Y diffèrent légèrement, alors $H(X)$ et $H(Y)$ doivent différer grandement.

Exemple : En identifiant les lettres majuscules à des nombres entre compris 0 et 25, comme nous l'avons déjà fait auparavant, nous pouvons calculer pour un message composé de n lettres :

a) la somme des n lettres modulo 26

b) la somme des n lettres, multipliée chacune par sa position dans le message, modulo 26

et obtenir ainsi une valeur de hachage composée des deux lettres résultantes. Voici un exemple : si le message est BONJOUR, alors

a) $B + O + N + J + O + U + R \pmod{26}$
$= 1 + 14 + 13 + 9 + 14 + 20 + 17 \pmod{26} = 88 \pmod{26} = 10 = K$

b) $1 \cdot B + 2 \cdot O + 3 \cdot N + 4 \cdot J + 5 \cdot O + 6 \cdot U + 7 \cdot R \pmod{26}$
$= 1 \cdot 1 + 2 \cdot 14 + 3 \cdot 13 + 4 \cdot 9 + 5 \cdot 14 + 6 \cdot 20 + 7 \cdot 17 \pmod{26}$
$= 1 + 28 + 39 + 36 + 70 + 120 + 119 \pmod{26} = 413 \pmod{26} = 23 = X$

Donc la valeur de hachage du message BONJOUR est KX. Tandis que vous pouvez vérifier que la valeur de hachage du message BONKOUR est LB : une petite modification dans le message d'origine mène bien à une valeur de hachage substantiellement différente.

Il est possible de faire de nombreux usages des fonctions de hachage :

- Celles-ci peuvent être utilisées pour vérifier qu'un message transmis ne contient pas d'erreur : si la valeur de hachage transmise ne correspond pas à celle attendue, on demande à l'expéditeur de retransmettre le message.

- Elle peuvent aussi être utilisées pour classer des grands jeux de données en économisant de la place. Au lieu d'enregistrer les données telles quelles, on enregistre leurs valeurs de hachage dans une *table de hachage* qui occupe moins de place en mémoire.

- Finalement, elles peuvent être utilisées à des fins cryptographiques, comme nous allons le voir.

Signature digitale

Revenons au problème évoqué plus haut : Alice veut transmettre un message à Bob et aussi lui donner un moyen de vérifier que c'est bien elle qui est l'auteure de ce message.

Une solution naïve serait la suivante : Alice transmet le message X à Bob, et calcule, avec une fonction de hachage H donnée, une valeur de hachage $H(X)$, qu'elle utilise comme signature du message. Cependant, ce système a le grave défaut de supposer qu'Eve ne connaît pas la fonction de hachage H ; il ne respecte donc pas le principe de Kerkhoffs vu au début de ce cours, qui dit qu'il faut toujours supposer qu'Eve connaît le système utilisé. Sous cette hypothèse, rien n'empêcherait Eve d'envoyer un message Y à Bob, et d'y apposer sa signature $H(Y)$ pour faire croire à Bob que ce message vient d'Alice. Il faut donc trouver autre chose.

Pour ce faire, nous allons réutiliser les idées du protocole RSA, mais à l'envers, en quelque sorte. Voici comment procéder :

1. *Alice* génère d'abord deux grands nombres premiers P et Q (distincts) et calcule $N = P \cdot Q$, qu'elle publie.
2. Puis elle choisit un autre nombre C entre 2 et $N-1$ tel que $\mathrm{PGDC}(C, \phi(N)) = 1$, qu'elle garde *secret*.
3. Elle calcule ensuite le nombre D tel que $C \cdot D \pmod{\phi(N)} = 1$ et *publie ce nombre*.
4. C'est ici qu'intervient la fonction de hachage H, publique elle aussi, dont on supposera que l'image est l'ensemble des nombres allant de 0 à $N-1$. Pour envoyer le message X, Alice calcule $H(X)$, puis $S = H(X)^C \pmod{N}$, et envoie finalement X et S à Bob. S est maintenant la vraie signature du message, qu'Eve ne peut pas contrefaire, comme nous allons le voir ci-dessous.
5. Pour vérifier que le message provient bien d'Alice, Bob calcule $H(X)$ et $S^D \pmod{N}$. Si ces deux nombres sont égaux, Bob a alors la garantie que le message vient bien d'Alice.

A nouveau, vérifions que tout ceci fonctionne bien :

1. Si le message X a bien été envoyé par Alice et que celle-ci a calculé la valeur de la signature S comme décrit ci-dessus, alors Bob trouve effectivement que $S^D \pmod{N} = H(X)^{C \cdot D} \pmod{N} = H(X)$ par le théorème d'Euler et le fait que $C \cdot D \pmod{\phi(N)} = 1$.
2. Pour contrefaire la signature d'Alice, Eve devrait connaître la valeur de C, mais elle ne connaît que N, D, X, S et la fonction H. Or elle ne sait pas factoriser N (ce qui lui permettrait de calculer $\phi(N)$, et donc de retrouver la valeur de C), ni résoudre le problème du logarithme discret $S = H(X)^C \pmod{N}$ (ce qui lui permettrait également de retrouver la valeur de C). Elle ne peut donc pas utiliser ce nombre C pour signer son propre message Y et ainsi faire croire à Bob que celui-ci provient d'Alice.

Remarque finale : Il faut évidemment éviter d'utiliser les deux protocoles décrits ci-dessus (RSA et DSA) à la suite l'un de l'autre, au risque de tout compromettre !