Principe et Algorithme Cryptographiques : examen

première session / documents et calculatrice autorisés / durée 2h / 2 pages

10 mai 2017

1 Paiement en ligne

La société AMOS et son prestataire WorldOffline offrent aux commerçants en ligne un mécanisme de paiement sécurisé. Lorsqu'il souhaite effectuer le paiement, l'acheteur envoie une requête HTTP POST à un serveur d'AMOS. Il est alors redirigé vers une page de paiement. Cette requête est composée de deux champs :

- `data` : un dictionnaire qui contient (entre autre) le montant de la transaction, le numéro de contrat du commerçant auprès de la banque, le numéro de carte de crédit de l'acheteur, sa date d'expiration et le cryptogramme au dos, ainsi qu'un identifiant unique de transaction.
- `seal` : SHA-256(`data` $\|$ K), où K est une clef secrète partagée à l'avance entre AMOS et le commerçant.

A l'issue de la transaction sur la page de paiement, AMOS envoie une requête POST à une adresse spécifiée par le commerçant. Cette requête contient deux champs :

- `data` : un dictionnaire qui contient (entre autre) l'identifiant unique de la transaction, et un code réponse (paiement accepté, refus par la banque, timeout, problème technique, etc.)
- `seal` : SHA-256(`data` $\|$ K).

▷ **Question 1 :** Quel est le rôle du champ `seal` ? Quelle primitive cryptographique réalise-t-il ?

Il s'agit d'un code d'authentification de message (MAC), une sorte de signature à clef secrète. Cela garantit l'authenticité des `data` : elles ne peuvent être modifiées pendant le transport, et elles proviennent forcément d'un expéditeur qui connaît K.

▷ **Question 2 :** De quelle manière est-ce que les acheteurs pourraient tricher si le `seal` n'était pas présent ?

Les acheteurs pourraient par exemple envoyer eux-même au site marchand le message disant « la transaction a réussi » au site marchand... même si en réalité la transaction a échoué. En principe, c'est la banque qui envoie ce message. Mais sans le `seal`, l'identité de l'expéditeur n'est pas garantie.

▷ **Question 3 :** La société AMOS vérifie soigneusement que l'identifiant unique de transaction n'a jamais été utilisé précédemment. De quelle manière le commerçant pourrait tricher si ce n'était pas le cas ?

La question est légèrement mal formulée. L'idée générale, c'est que les identifiants unique de transaction empêchent les attaque par *rejeu*. Par exemple, un petit malin pourrait intercepter le message de demande de paiement envoyé par le marchand à la banque, puis le ré-emettre plusieurs fois (au bénéfice du marchand, et au préjudice du client).
En fait, une fois qu'il possède les coordonnées bancaires de l'acheteur, rien n'interdit au marchand d'effectuer plein de transactions en son nom !

▷ **Question 4 :** Quel problème de sécurité majeur se pose si toutes ces requêtes sont envoyées « en clair » sur le réseau ?

Les coordonnées bancaires du client, qui sont suffisantes pour permettre le paiement en-ligne, sont révélées à tous ceux qui écoutent la connection (fournisseur d'accès internet, employés de maintenance du datacenter, etc.)

▷ **Question 5 :** En réalité, ces requêtes sont envoyés en HTTPS. Expliquez par quel mécanisme ceci garantit à l'acheteur qu'il envoie son numéro de carte bancaire à une banque et pas à un serveur pirate.

Le protocole HTTPS prévoit une authentification à base de certificats. La banque présente un certificat qui associe son nom de domaine et sa clef publique. Ensuite, elle réalise une signature avec sa clef secrète pour démontrer son identité à l'acheteur.

▷ **Question 6 :** L'outil `openssl` révèle que la connection HTTPS utilise DH+RSA+SHA384+AES256CBC. Expliquez le rôle de chacun de ces algorithmes.

- DH : il s'agit de l'échange de clef Diffie-Hellman, utilisé pour établir une clef de session secrète partagée, qui servira à chiffrer toute la suite avec un algorithme de chiffrement symétrique.
- RSA : utilisé pour effectuer la signature des parts de l'échange de clef Diffie-Hellman, afin d'éviter les attaques par le milieu.
- SHA384 : la fonction de hachage utilisée pour la signature (et aussi pour calculer des MACs des données qui transitent ensuite).
- AES-256-CBC : l'algorithme de chiffrement symétrique utilisé pour réaliser le « tunnel » chiffré une fois la session établie.

▷ **Question 7 :** Pourquoi ne pas avoir utilisé `seal` $= \mathsf{SHA\text{-}256}(K \parallel \texttt{data})$?

Probablement pour éviter la *length-extension attack* (cf. le poly).

2 Retour sur les TPs

Le protocole STP est un protocole jouet créé exprès pour les TPs de PAC. Il permet à un client de s'identifier auprès d'un serveur avec un mot de passe. Voici comment il fonctionne :

1. Le client envoie son `username` au serveur.
2. Le serveur répond avec un `nonce` aléatoire.
3. Les deux calculent la clef de session $K =$ `password` $\parallel$ `nonce`.
4. Le client envoie une requête prédéfinie, toujours la même, chiffrée avec K.
5. La session du client est ouverte, et les échanges qui suivent sont chiffrés avec la clef K.

Dans le TP, la dernière étape se réalise en envoyant le dictionnaire `{'url': '/bin/stp/gateway', 'method': 'GET'}` chiffré avec K.

▷ **Question 8 :** Expliquez comment un adversaire passif qui espionne une session du protocole peut effectuer une attaque par dictionnaire « hors-ligne » sur le `password`.

Une fois qu'on a obtenu le `nonce`, on peut, pour chaque `password` appartenant à un dictionnaire, calculer K puis tenter de déchiffrer la première requête avec K. Si ça donne les données attendues (fournies par l'énoncé), c'est que ce `password` est manifestement le bon. Ceci ne nécessite pas d'interaction en-ligne avec les participants, autre que la capture d'une session du protocole.

▷ **Question 9 :** Le dictionnaire (anglais) fourni dans le TP contient $\approx 2^{16}$ mots, qui font tous 9 lettres (pour simplifier). Dans chacun des cas suivants, estimez le temps que prendrait une recherche exhaustive sur le mot de passe. Indiquez si cela vous semble réalisable par un particulier sans moyens importants.

a. Mon mot de passe est formé de 4 mots choisis aléatoirement dans le dictionnaire.

Il y a $\left(2^{16}\right)^4 = 2^{64}$ combinaisons de 4 mots. Admettons (c'est très optimiste) qu'un particulier soit capable d'en tester 2^{32} par seconde, vu qu'un CPU moyen fonctionne à environ 2^{32} Ghz. Il va donc lui falloir 2^{32} secondes pour effectuer la recherche. Ceci n'est pas réaliste : il y a 86400 secondes par jour, soit environ 2^{16}. Ca fait donc 2^{16} jours, soit 180 ans !

b. Mon mot de passe est formé d'un mot aléatoire suivi de 4 chiffres aléatoires.

Il y a $2^{16} \times 10000 \approx 2^{30}$ combinaisons de cette sorte. Sous les mêmes hypothèses que précédemment, la recherche exhaustive prendrait une seconde : ça n'est pas irréaliste !

c. Mon mot de passe est un mot de choisi aléatoirement, et chacune des lettres a été mise en majuscule avec probabililité 1/2.

Comme on a supposé que les mots de passe font 9 lettres, il y a 2^9 combinaisons de minuscules/majuscules possible pour chaque mot. Cela fait donc $2^{16} \times 2^9$ mots de passes différents, soit 2^{25}, bien trop peu ! La recherche exhaustive est bien possible.

3 Partage de secret

Cet exercice est consacré au schéma de partage de secret de Shamir. Il permet de partager un *secret* en autant de *parts* qu'on le souhaite. Lors du partage, un *seuil* T est fixé : avec moins de T parts, on n'obtient rien, tandis qu'avec T parts on peut reconstituer le secret. L'acteur qui effectue le partage est nommé le *dealer*.

Dans tout le processus, on suppose qu'on grand nombre premier p (occupant k bits) a été fixé et qu'il est connu de tous. Pour partager un secret S (qui est un nombre S avec $0 \leq S < p$) avec un seuil de $T \geq 2$, le *dealer*

choisit uniformément au hasard $T-1$ élements de $\mathbb{Z}_p$ notés $A_1, \ldots, A_{T-1}$. La i-ème part ($i \geq 1$) s'obtient en calculant :

$$P_i = \left(i^{T-1} \cdot A_{T-1} + \cdots + i^2 \cdot A_2 + i \cdot A_1 + S\right) \ \% \ p.$$

▷ **Question 10 :** Quelle est la complexité, en fonction de k et T, du calcul d'une part ?

En se débrouillant bien, on peut s'en tirer avec deux multiplications et une addition dans $\mathbb{Z}_p$ par terme de la formule ci-dessus. Là-dedans, la multiplication coûte $O(k^2)$. La complexité totale est donc en $O(Tk^2)$.
Si on s'y prend moins bien (c.a.d. si on calcule i^{T-1} en ignorant qu'on possède déjà i^{T-2}), alors la complexité monte à $O(T^2k^2)$.

▷ **Question 11 :** Quel gros problème y a-t-il si on autorise la part avec $i = 0$?

$P_0 = S$. Elle révèle le secret !

▷ **Question 12 :** Que se passe-t-il si le *dealer* publie accidentellement les coefficients $A_1, \ldots, A_{T-1}$?

Le secret peut alors être déduit de n'importe quelle part isolée. En effet, on aura alors :

$$P_i - \left(i^{T-1} \cdot A_{T-1} + \cdots + i^2 \cdot A_2 + i \cdot A_1\right) \equiv S \mod p$$

Le mécanisme offre la *sécurité inconditionnelle*, comme on va le démontrer.

▷ **Question 13 :** Supposons qu'on dispose d'une seule part P_i d'un secret partagé. Montrez que pour tout secret S, il existe des valeurs des coefficients $A_1, \ldots, A_{T-1}$ qui produisent la même valeur de P_i.

Voici un algorithme qui répond à la question posée.
- INPUT : S, P_i.
- OUTPUT : $A_1, \ldots, A_{T-1}$.
- Fixer $A_2 \leftarrow 0, \ldots, A_{T-1} \leftarrow 0$.
- Calculer $A_1 \leftarrow i^{-1}\,(P_i - S) \bmod p$.

Il est facile de vérifier qu'à la fin, on a bien la bonne valeur de P_i.

▷ **Question 14 :** Concluez-en qu'un adversaire qui aurait une puissance de calcul énorme (qui serait par exemple capable d'effectuer 2^k opérations) ne pourrait pourtant pas retrouver S à partir de P_i.

C'est le même argument que pour la sécurité du masque jetable : on vient de voir que P_i pourrait être une part de n'importe quel secret (en faisant varier les coefficients (A_i)), et aucun calcul ne peut permettre de faire la différence entre une part du secret « Larguez la bombe » et une part du secret « rentrez à la base ». En fait, les coefficients A_i, choisis complètement au hasard, jouent le rôle d'un masque jetable.

Le résultat ci-dessus s'étend aussi au cas où l'adversaire possède $T-1$ parts, mais ça on va l'admettre. On va maintenant voir comment le secret peut être reconstitué avec T parts.

▷ **Question 15 :** Supposons que $T = 2$. Expliquez comment, avec deux parts P_i et P_j ($i \neq j$), on peut reconstituer le secret S.

On a $P_i \equiv iA_1 + S \bmod p$ et $P_j \equiv jA_1 + S \bmod p$. Par conséquent, $P_i - P_j \equiv (i-j)A_1 \bmod p$, et donc on peut retrouver A_1 en calculant : $A_1 \equiv (P_i - P_j)(i-j)^{-1} \bmod p$. Après, retrouver S est facile (cf. correction de la question 12).

▷ **Question 16 :** Expliquez comment le processus se généralise lorsque $T > 2$. Comment la complexité évolue-t-elle en fonction de T ?

Le problème se ramène toujours à la résolution d'un système de T équations linéaires en T inconnues modulo p. Les inconnues sont $A_1, \ldots, A_{T-1}$ et S. Il se trouve que ce système a toujours une et une seule solution (la preuve de cette affirmation nous emmenerait trop loin).
On peut le résoudre par la méthode que vous avez apprise au collège/lycée, le *pivot de Gauss*. Ceci nécessite $O(T^3)$ opérations arithmétiques, donc le coût total est en $O(k^2T^3)$.

Ce système de partage de secret possède une propriété de morphisme. On partage un premier secret S avec les coefficients $A_1, \ldots, A_{T-1}$, ce qui donne les parts $P_1, P_2, \ldots$. Ensuite, on partage un deuxième secret S' avec d'autres coefficents $A'_1, \ldots, A'_{T-1}$, ce qui donne les parts $P'_1, P'_2, \ldots$.

▷ **Question 17 :** Montrez que les parts $(P_i + P'_i)$ % p correspondent à un partage du secret $(S + S')$ % p. Quels seraient les coefficients correspondants ?

Il suffit de l'écrire pour voir que c'est vrai :

$$P_i \equiv i^{T-1} \cdot A_{T-1} + \cdots + i^2 \cdot A_2 + i \cdot A_1 + S \mod p$$
$$P'_i \equiv i^{T-1} \cdot A'_{T-1} + \cdots + i^2 \cdot A'_2 + i \cdot A'_1 + S'$$

Donc, en sommant ces deux congruences on trouve :

$$P_i + P'_i \equiv i^{T-1} \cdot (A_{T-1} + A'_{T-1}) + \cdots + i^2 \cdot (A_2 + A'_2) + i \cdot (A_1 + A'_1) + (S + S') \mod p.$$

On voit donc apparaître que $P_i + P'_i$ correspond au partage de $S + S'$ avec les coefficiens $(A_1 + A'_1), \ldots, (A_{T-1} + A'_{T-1})$.

En cette période électorale, on va voir une application du partage de secret aux élections. Imaginons un référendum où les citoyens peuvent voter « 0 » ou « 1 ». Le résultat de l'élection (qu'on note R) est le nombre de votes « 1 ». Il y a N bureaux de vote, numérotés de 1 à N, et chaque bureau connaît tous les électeurs. Pour voter, un électeur partage son vote (0 ou 1) en N parts, avec un seuil de N, et envoie par un canal sécurisé une part à chaque bureau (la i-ème part au i-ème bureau).

▷ **Question 18 :** Imaginons que le résultat R de l'élections soit un secret partagé entre les bureau de vote. Expliquez comment le i-ème bureau de vote peut calculer la i-ème part du partage de R à partir des votes.

La somme des votes des votants donne R. La somme des parts des votants donne la part de la somme, c'est-à-dire la part de R. Bilan : le bureau de vote doit calculer la somme de tout ce qu'il reçoit modulo p.

▷ **Question 19 :** Expliquez comment, une fois le scrutin terminé, les N bureaux de vote peuvent obtenir le résultat de l'élection.

Les N bureaux effectuent la recombinaison de leurs N parts de R, et ils vont pouvoir reconstituer R, le résultat de l'élection.

▷ **Question 20 :** Supposons qu'un des bureaux est honnête et refuse de publier les parts qu'il reçoit des électeurs. Les autres bureaux, qui sont tous malhonnêtes et qui peuvent collaborer entre eux, peuvent-ils apprendre pour qui un électeur a voté ?

Non. Sous l'hypothèse que $T-1$ parts ne permettent pas de reconstituer le secret, avec $N-1$ parts il en manque encore une pour atteindre le seuil fixé à N lors du partage de chaque part.

▷ **Question 21 :** Ce protocole, tel qu'il est décrit, souffre de nombreux défauts. Parmis ceux-ci, un électeur mécontent, ou un bureau de vote malhonnête, peuvent gravement fausser l'issue de l'élection. Expliquez comment ils pourraient s'y prendre (il est possible d'y remédier... mais c'est assez compliqué !).

D'abord, les électeurs peuvent tricher en envoyant des partages de -10 ou $+1000$ pour pousser le résultat global dans le sens qu'ils ont envie... Ensuite, un bureau de vote pourrait refuser de participer à la reconstruction du résultat de l'élection, qui resterait alors caché. Plus subtil, un bureau de vote pourrait fournir une part aléatoire lors de la reconstruction, aboutissant à un résultat R aléatoire lui aussi, modulo p.