M1 Informatique # Protocoles réseaux

TD n°11

Sécurité des réseaux

Hachage cryptographique

HD5, SHA256, Argon2, etc.

- $U_{naire} \rightarrow \{0,1\}^n$
- $binaire\ \{0,1\}^*$
- $M \longmapsto h(M)$
- Inversion $h(M)_+ \longmapsto M$ très difficile

Minage bitcoin

while (h(M) ne commence pas par 0^{17*8})
then M

Quand on parle de cryptographie :

1. Crypter : pour que le message soit illisible.
2. Signer le message (authentification) : pour que le message soit infalsifiable, y compris pour l'émetteur.
   HMAC
3. Vérifier l'intégrité : pour que le message soit infalsifiable.
   m25sum(M), sha256sum(M), M = le message

Quelle est la façon la plus simple de hacher un message ?

Notre outil c'est HMAC.

Si on sait partager un secret k, une certaine chaine comme par exemple « arerty », on envoie : $(M, sha256sum(k.M))$

Exercice 1

Lorsque Alice a ouvert un compte à la banque de Bernard, ils se sont échangés une clé privée k.

Lorsque Alice désire faire un virement, elle signe le message avec un HMAC de clé k : si

$$m = \text{« } Transférer\ 100zł\ sur\ le\ compte\ de\ Chloé \text{ »}$$

alors Alice envoie à Bernard le message $(m, HMAC_k(m))$.

On suppose que le protocole HMAC est invulnérable à la contrefaçon :
Chloé (qui ne connaît pas k) ne peut pas générer un HMAC valide
(étant donné m' elle se sait pas calculer $HMAC_k(m')$),
elle peut juste rejouer[1] des HMAC qu'elle a capturés.

On suppose que Chloé peut intercepter, rejouer ou corrompre les messages envoyés d'Alice [2].
On suppose aussi que le canal n'est pas fiable : un message peut être perdu, avant ou après que Chloé l'ait intercepté.

---

[1] C'est-à-dire envoyer une deuxième fois le même contenu $(m, HMAC_k(m))$ qu'elle a capturé auparavant.

[2] Le facteur est amoureux de Chloé.

HMAC pour l'authentification message-par-message

HMAC: Hashed Message Authentication Code

Pour créer un HMAC, on ajoute une clé secrète au message en clair d'origine. On hache la chaine de bits combinée au moyen de MD5 ou de SHA1. On obtient alors le HMAC, que l'on joint aux messages sortants pour l'authentification.

Clé	Texte en clair d'origine

↓ **Hachage avec MD5, SHA1, etc.**

HMAC

HMAC (Hashed Message Authentication Code)

↓ **Ajouté au texte en clair avant la transmission**

HMAC	Texte en clair d'origine

Source : Sécurité des systèmes d'information et des réseaux Panko, Raymond R.

1#

Chloé peut-elle ajouter des chiffres au montant à transférer ?

Ajouter 3 « 0 » par exemple ? Elle connait pas $\boldsymbol{k}$, donc si elle essai de modifier $\boldsymbol{m}$, elle peut pas concaténer le $\boldsymbol{m}$ modifié avec le $\boldsymbol{k}$ et le hacher.

2#

Chloé fait une copie du message d'Alice et la réémet plus tard à Bernard. Que se passe-t-il ?

C'est possible. Si on renvoie 1000 fois ce message elle va pouvoir vider le compte d'Alice à son profit.

3#

Les techniques de *channel binding*, où le HMAC contient un identificateur de l'émetteur ou du récepteur, résolvent-elles le problème ?

Channel binding consiste à authentifier un canal. Quesque on peut faire pour authentifier un canal ? Imaginer que le secret $\boldsymbol{k}$ est pas partager que par Alice et Bernard, mais aussi

par Damien. Damien pourra faire des faux messages ? Là, à priori, oui. Comment faire pour que Damien ne puisse pas fabriquer de faux messages ? Damien il à u la clé secrète (il à volé la clé), mais il peut pas intercepter (saisir, capturer) le message. Donc, maintenant Damien connait le secret, donc il peut faire $\boldsymbol{sha256(k.m)}$ et envoyer à Bernard, mais quand Bernard il reçoit le message, il peut pas exécuter l'ordre banquer, car il manque l'expéditeur du message, numéro de compte et autres info qui permette d'authentifier.

$$(\boldsymbol{M, sha256sum(k.''Alice''.M)})$$

La technique du challenge

Ca peut se traduire par « défi ». Comment Bernard peut défier Alice qu'elle est vraiment Alice et pas Damien ?

Challenge : Bernard envoie M_x à Alice, et Alice doit envoyer $\underbrace{sha256sum(k.M_x)}_{\text{Alice crypte et s'authentifie}}$.

Maintenant, si Alice envoie le message suivant ca garantie que c'est bien Alice (Damien peut pas envoyer ca) :

$$\boldsymbol{('' \ldots 100zł \ldots'', sha256sum\ k, M_x, (\ '' \ldots 100zł \ldots''))}$$

Et donc est-ce que ça résolu le problème ? **Oui, si challenge à chaque message.**

Mais, normalement, c'est pas utiliser à chaque message, c'est plutôt pour les clés de session.

KS clé de session

$$KS = sha256(k.M_x."Alice"."Bernard")$$

Puis cryptage symétrique avec KS $\mathbf{Crypt}()$
Cryptage symétrique → AES : Algo de cryptage symétrique.

$\mathbf{Crypt(KS, M)}$ → Alors pour Chloé le message est illisible car elle a pas k → elle peut pas décrypter KS .....(à compléter)..... Donc, solution : on numérote les messages pour les rendre uniques.

Méthode pour rendre un message symétrique

(Pour que 2 messages diffèrent de la liste ne pourrait pas être rejouer)

1. Numéro séquence
2. Timestamp avec la date (Idée pas très bonne) – Inconvénient : Bernard et Alice on pas forcement la même horloge.
3. Nonce (encore une idée pas très bonne) – Numéro aléatoire sur beaucoup de bits (un numéro qui n'a aucun sens mais est unique). Problème (pas un problème de sécurité, car la sécurité est assurer, mais un problème d'implémentation) : Quelle est la condition pour que Bernard accepte ou rejette un message ? Il doit se rappeler de tous les précédents « nonce » , donc Chloé risque d'envoyer un message d'il y a 3 ans...

Pour pallier à ce problème,
Alice numérote chacun de ses messages, et inclut le numéro du message dans la signature :

$$m = \text{« 42 : transférez 100zł sur le compte de Chloé »}$$

et Alice envoie à Bernard le message $(m, HMAC_k(m))$.

Bernard maintient le numéro du dernier message reçu,
et ignore tous les messages dont le numéro de séquence
n'est pas strictement supérieur au dernier.

4#

Que se passe-t-il si Chloé essaie de rejouer un message ?

On peut pas rejouer le message.

Bernard maintient le numéro du dernier message reçu, et ignore tous les messages dont le numéro de séquence n'est pas strictement supérieur au dernier.

5#

Que se passe-t-il si un message d'Alice est retardé et il se fait doubler par le message suivant ?

Si le deuxième message arrive plus vite que le premier ?

Si Bernard reçoie le message 43 et ensuite le message 42, il accepte 43 mais ignore 42.

Donc, le message 42 va être refuser par Bernard, quelle outil on peut imaginer ?

On pourrait dire qu'une fois qu'il attend le message 42, si il reçoit par exemple le message 43, alors il le rejette aussi. Quelle est le problème avec ça ? Si il y a une perte, alors Alice continue d'augmenter les numéros de séquences.

Alice envoie : $\underbrace{42}_{\downarrow}$ $\underbrace{\cancel{43}}_{Perdu}$ $\underbrace{44}_{\downarrow}$

$Ack(42)$ $Ack(44)$

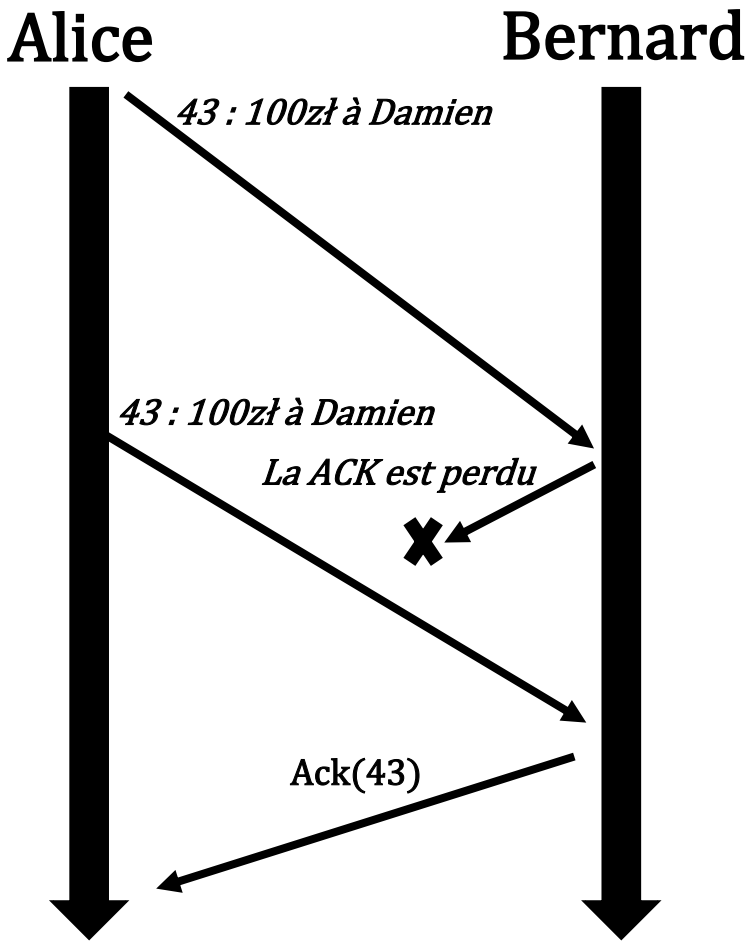
Dans ce cas-là : quelle est le problème si Alice envoie 43 ?

Inefficace car faudrait envoyer : $\underbrace{43}_{\downarrow}$ $\underbrace{44}_{\downarrow}$

$Ack(43)$ $Ack(44)$

Imaginer un scénario : on incrémente les numéros de séquences et portant ça marche pas ?

.



6#

Que se passe-t-il si Bernard oublie le dernier numéro de séquence ? Si Alice l'oublie ? Si les deux l'oublient ? Proposez une solution à ce problème.

Si Alice oublie son numéro de séquence :
Alice le demande à Bernard. Bernard, pour ce rassurer, Challenge Alice. Il va demander une authentification forte pour savoir que c'est vraiment elle.

Si les 2 oublies :
La solution c'est le double challenge. Chaque-un challenge l'autre : A challenge B, B challenge A ⟶ Channel binding.

Exercice 2

James Bond (007) quitte le MI6 avec une collection de $one - time\ pads$, des suites aléatoires de 256 octetschacune :

$$S_0 = 6d\ 92\ ae\ 9b\ 66\ fb \ldots$$
$$S_1 = a7\ 0a\ 63\ 04\ 6a\ fe \ldots$$
$$\ldots$$

Lorsqu'il désire envoyer un message m_0 de longueur inférieure à 256 octets, Bond le complète avec des 0 pour qu'il ait une longueur de 256 octets exactement ; il obtient alors un message m'_0 de longueur 256. Il calcule ensuite

$$c_0 = m'_0\ XOR\ S_0$$

et il envoie c_0 au MI6. Lorsqu'il reçoit le message c_0, M. calcule

$$m''_0 = c_0\ XOR\ S_0$$

XOR est l'opération ***ou exclusif***, notée « ^ » en C ou ⊕ en Mathématiques. C'est une opération classique sur les bits :

- 0 ⊕ 0 = 0
- 0 ⊕ 1 = 1
- 1 ⊕ 0 = 1
- 1 ⊕ 1 = 0

Remarquez les propriétés suivantes :

- a ⊕ b = 0
- a ⊕ b ⊕ b = a

Comme l'application du ***ou exclusif*** deux fois avec la même valeur redonne la valeur initiale, le chiffrement et le déchiffrement utilisent exactement le même programme :

- M ⊕ K = C
- C ⊕ K = M

Source : Cryptographie appliquée : protocoles, algorithmes et codes source en C / Bruce Schneier

1#

Montrez que $m''_0 = m'_0$. Conclusion ?

$$m''_0 = c_0 \; XOR \; S_0$$

$$c_0 = m'_0 \; XOR \; S_0$$

Comme l'application du ***ou exclusif*** deux fois avec la même valeur redonne la valeur initiale, le chiffrement et le déchiffrement utilisent exactement le même programme :

- M ⊕ K = C
- C ⊕ K = M

$$\Rightarrow m'_0 = c_0 \; XOR \; S_0$$

Le XOR est associatif

$$(\mathbf{A} \oplus \mathbf{B}) \oplus \mathbf{B} = \mathbf{A} \oplus (\mathbf{B} \oplus \mathbf{B}) = \mathbf{A}$$

$$\underbrace{m'_0 \oplus S_0}_{C_0} \oplus S_0$$

$C_0 \longrightarrow$ Ce que circule sur le réseau

Conclusion

1. Cryptage
2. Authentification
3. Intégrité

Le cryptage est bon ?

On obtient un message facilement décryptable ou indécryptable ?

Le **XOR** de n'importe quoi est une chaine aléatoire ⇒ a tous les propriétés de l'aléatoire. Donc : indéchiffrable, le cryptage est OK.

Authentification

Il faut avoir S_0 Donc : authentification PAS OK.

Intégrité

Si il y a des bits corrompus ⇒ avec XOR ils restes corrompus

2#

Francisco Scaramanga intercepte le message c_0.
Que peut-il déduire à propos du message d'origine ?

Est-ce-que on sait que c'est James Bond qui la envoyé ?

Rien. L'aléatoire ne donne aucun information. On sait rien du tous quand on a le message.

3#

Pourquoi Bond a-t-il pris avec lui toute une collection de clés secrètes ?

(Indication : considérez ce que peut faire Scaramanga s'il intercepte deux messages chiffrés avec la même clé.)

Quelle est la faiblesse si j'ai :

$$c_0 = m'_0 \; XOR \; S_0$$

$$c_1 = m' \; XOR \; S_0$$

C'est quoi la fonction qui donne 2 parties identiques entre 2 messages ? **XOR**

$$\underbrace{c_0 \oplus c_1}_{\substack{XOR\ de\ 2 \\ messages \\ chiffrées}} = (m'_0 \oplus S_0) \oplus m_0 \oplus S_0) = \underbrace{m'_0 \oplus m'_1}_{\substack{XOR\ de\ 2 \\ messages \\ d'origine}}$$

Le **XOR** de deux messages contient des info utiles à un cryptoanalyste.

4#

Quel est le problème principal de ce protocole ?

Problème d'implémentation : avoir plains de time pads.

Pour chaque « seed » k,
un générateur de nombres aléatoires cryptographiques produit une suite d'octets S_k.

Le chiffrement par flots consiste à utiliser la suite S_k comme $one - time\ pad$

5#

Les vieilles versions de Microsoft Word utilisaient $RC4$, un algorithme de chiffrement par flots, pour chiffrer les documents.

La clé secrète k était dérivée de façon déterministe d'un mot de passe fourni par l'utilisateur.

Trouvez la faille.

Chiffrement par flots

Un générateur de nombres aléatoires cryptographique produit un aléatoire qui est pas vraiment un aléatoire...

```c
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
int main() {
    srand( time(NULL) );
    int x;

    x = rand();
    printf("%d", x);
    return 0;
}
```

Si on connait l'heure de l'exécution, on peut savoir c'est quoi l'aléatoire générer

⇒

Pas fiable.

Pour un bon aléatoire :

- /dev/random
- /dev/urandom

Si on crypte 2 documents Word, on va faire quoi ?

Une utilise le même mot de passe tous les jours... (à compléter)
Donc, quelqu'un qui a pas le mot de passe, quesque il peut faire ?

Question éthique

De plus que l'algorithme ..... de Microsoft est sécurisé ⇒ on peut pas le challenger.

En cryptographie, il est préférables d'avoir des choses à code ouvert.

Exercice 3 Confidentialité persistante et homme au milieu

Arthur et Bérénice s'échangent des messages.

Ils se sont échangés depuis longtemps une clé secrète x,

et ils utilisent x pour chiffrer tous les messages qu'ils s'échangent.

Einstein, le chat d'Arthur, a piraté la« box » d'Arthur,
un routeur par lequel passe tout le traffic, et sur lequel il peut exécuter $tcpdump$.

1#

On suppose que le cryptosystème employé par Arthur et Bérénice garantit la confidentialité :

on ne peut pas déchiffrer un message sans connaître la clé secrète.
Einstein, le chat d'Arthur, peut-il lire les messages échangés ?

Par définition, on à un système qui garantit la confidentialité : donc non.

Einstein, toutefois, est très patient :
il se contente de stocker les messages chiffrés qu'il a intercepté.

Après de longs mois à feindre le gentil chat et à marcher sur le clavier,
il a réussi à obtenir la clé secrète..

2#

Einstein peut-il maintenant déchiffrer les messages qu'il a interceptés un an plus tôt ?

Oui, dit qu'on à la clé secret on peut décrypter tous les messages.

Depuis quelques semaines, Arthur et Bérénice ont mis à jour leur protocole.

Ils se sont mis d'accord sur une fonction F (connue d'Einstein) qui vérifie les propriétés suivantes :

- pour tous entiers a et b, $F(F(42,a),b) = F(F(42,b),a)$;
- pour entier n, il est infaisable de retrouver n à partir de $F(42,n)$.

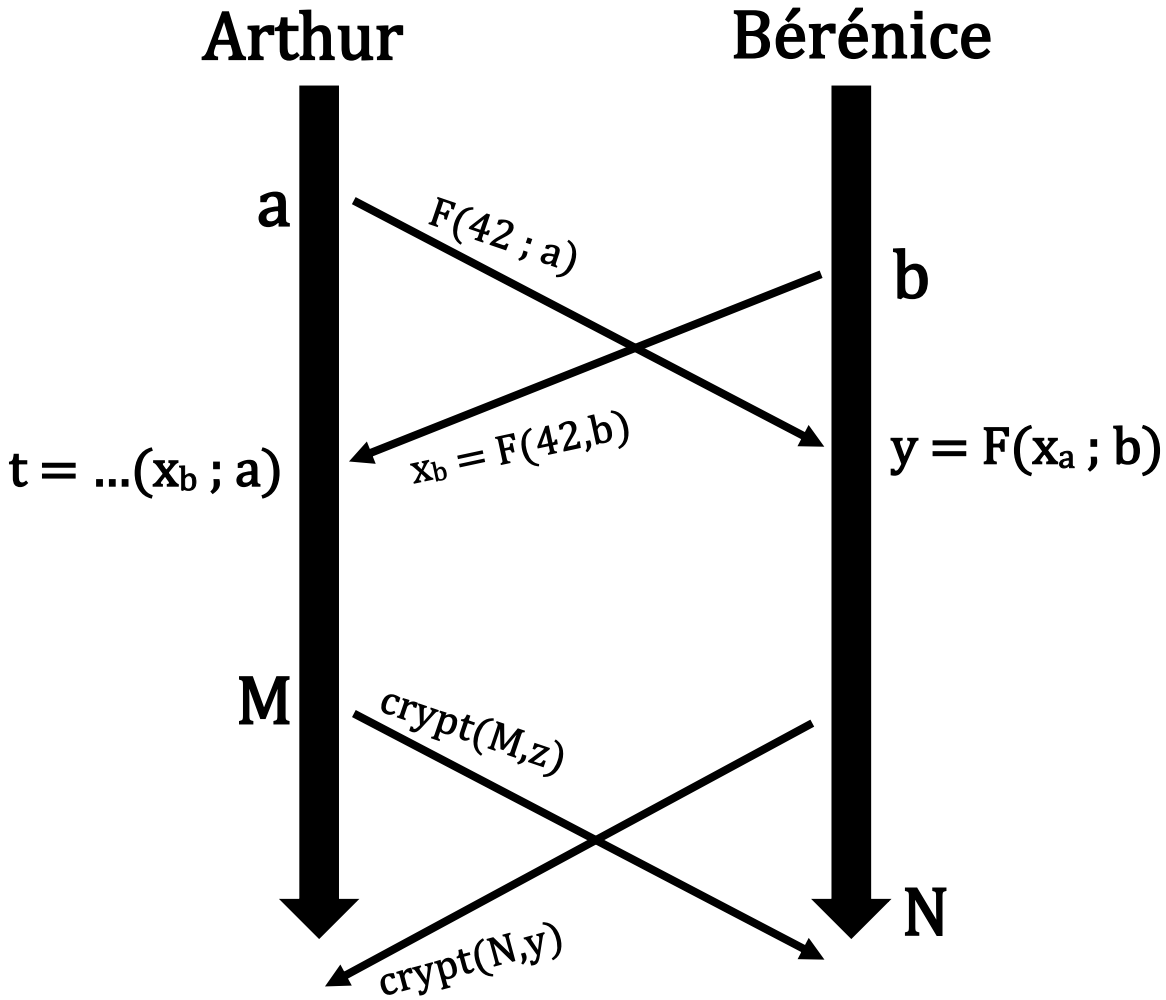
Lorsqu'ils veulent communiquer, ils procèdent ainsi :

- Arthur génère un nombre a au hasard ;
- Bérénice génère un nombre b au hasard ;
- Arthur envoie $x_a = F(42,a)$ à Bérénice ;
- Bérénice envoie $x_b = F(42,b)$ à Arthur ;
- Bérénice reçoit x_a et calcule $y = F(x_a,b)$;
- Arthur reçoit x_b et calcule $z = F(x_b,a)$.

Arthur chiffre/déchiffre ses message avec z et Bérénice avec y.
Après la fin de l'échange, ils effacent a, b, y et z.

3#

Montrez que $y = z$ à la fin de l'échange.
Conclusion ?



$t = T(x_b, a) = F(\ (F(42, b), a\ )$

$y = F(x_a, b) = F(\ (F(42, a), b\ )$ ‖

La clé de session

- Un algo semblable : Diphie - Hellman

4#

On suppose qu'Einstein se limite à des attaques passives : il ne peut pas modifier les données. Peut-il lire les messages ?

Pourquoi est-ce-que Einstein ne peut pas lire les messages ? alors que Einstein connait $\boldsymbol{F}$, il connait $\mathbf{42}$, mais il peut pas calculer ni $\boldsymbol{z}$ ni $\boldsymbol{y}$ (il connait pas $\boldsymbol{a}$),

A partir de $\boldsymbol{F}(\mathbf{42}, \boldsymbol{b})$ on peut pas avoir $\boldsymbol{b}$

non- car $\boldsymbol{F}$ non inucisible.