Conception des protocoles cryptographiques

Sécurité Informatique (Licence 3)

Abdelmalik Bachir

Généralités et définitions

Algorithme cryptographique : c'est une séquence d'instructions généralement s'exécutant au sein d'*une seule entité* pour réaliser un objectif de sécurité donné.
Ex : algorithme de cryptage AES, algorithme de cryptage RSA, …

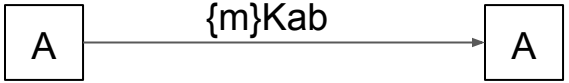
Protocole cryptographique : c'est une séquence d'étapes spécifiant les actions respectives devant être réalisées par *deux entités (ou plus)* pour remplir un objectif de sécurité donné.
Ex : protocole d'échange de clés Diffie Hellman)

Un protocole cryptographique *utilise un ou plusieurs algorithmes* de cryptographie

Généralités et définitions

Les algorithmes de cryptographie ne suffisent pas pour garantir une propriété cryptographique donnée entre deux entités (ou plus).
Un algorithme tout seul ne peut pas garantir la confidentialité d'un message.

Ex : A envoie à B un message m crypté avec un algorithme symétrique utilisant une clé Kab



Si la clé Kab n'est pas secrète aucun algorithme ne peut garantir la confidentialité

Garantir la confidentialité de la clé Kab est le rôle d'un protocole cryptographique

Types des protocoles cryptographiques

- Protocoles d'échange de clés (Ex : protocole de Diffie Hellman)
- Protocoles d'authentification
  - Authentification de l'origine des données
  - Authentification de l'entité homologue (ou identification)
- Protocoles combinant authentification et échange de clés
- Autres protocoles :
  - Vote électronique
  - Partage de clé de groupe (group key agreement)

Notations

- Kab désigne une clé secrète partagée entre a et b (algo symétrique)
- PKa désigne une clé publique asymétrique de a (algo asymétrique)
- SKa désigne une clé privée asymétrique de a (algo asymétrique)
- m désigne un message
- {m}Kab désigne m chiffré avec Kab
- {m}PKa désigne m chiffré avec PKa
- {m}SKa désigne m chiffré avec SKa
- H(m) désigne un condensé calculé sur m avec la fonction de hachage h
- Hk(m) désigne un MAC calculé sur m avec la fonction de hachage Hk paramétrée avec la clé k
- . désigne l'opération de concaténation (Ex: m.H(m) m concaténé a H(m))

Notations

- Les entités communicantes :
  - A : Alice
  - B : Bob
  - C : attaquant
  - S : serveur
- Les messages (échange entre les entités)
  - Exemple: A -> B : m (A envoie à B un message m)
  - …
- Les protocoles (séquence d'envoie de messages)
  - Exemple:
    i. A -> B : {m}PKb (A envoie à B un message m crypte par la clé publique de B)
    ii. B -> A : m (B envoie à A un message m)

Notations

- Les nonces (oNly ONCE) : nombres aléatoires générés par les entités
  - ra : nonce généré par a
  - rb : nonce généré par b,
  - …
- Les numéros de séquences
  - na : numéro de séquence maintenu par a (incrémenté à chaque envoi)
  - nb : numéro de séquence maintenu par b (incrémenté à chaque envoi)
  - ...
- Les estampilles (timestamp : marqueur de temps)
  - ta : estampille au niveau de a
  - tb : estampille au niveau de b
  - ...

Hypothèses sur l’attaquant

Dans l'étude de la sécurité des protocoles de communication, il faut fixer les hypothèses sur les attaquants car si on suppose que l’attaquant a des capacités hors normes, il pourrait casser tout système. Dans ce cours, nous supposons que :

- C peut écouter les messages échangés
- C peut bloquer les messages
- C peut rediriger les messages
- C peut enregistrer les messages
- C peut rejouer les messages
- C ne sait pas déchiffrer (sans avoir la clé) dans le temps d'une session

Protocoles d'échange de clés

Exemple d'échange de clé en utilisant un système asymétrique

- M1: b->a : PKb
- M2: a->b : {Kab}PKb

Ce protocole ne fonctionne correctement que si la clé PKb est vraiment celle de b

Sinon, une attaque du type Man In The Middle est possible

- M1: b->c/a: PKb
- M2: c/a->b: {Kcb}PKb

Dans ce cas, b a établi une clé symétrique avec l'attaquant mais pas a

Protocoles d’authentification de l’origine

L’objectif de ces protocoles est de garantir à b que le message a été créé par a. On peut garantir cette propriété avec des algorithmes symétriques ou asymétriques.

Avec un algorithme à clés symétriques (HMAC)

- M1: a->b : a.m.Hk(m)

Avec un algorithme à clés symétriques (signature)

- M1: a->b : a.{m}SKa

Attention : ce protocole ne protège pas contre le rejeu. Pour contrer cette attaque, il faut ajouter un paramètre pour assurer la fraîcheur : estampille, numéro de séquence, nonce envoyé dans un message précédent.

Protocoles d'authentification d'entité

L’objectif de ces protocoles est de garantir à b d’authentifier a. Il y a deux formes:

- Authentification faible : mots de passe fixes
- Authentification forte : protocoles défi-réponse

Dans certaines situation, l’authentification faibles n’est pas suffisantes. Par exemple, si vous effectuez un achat en ligne d’une certaine somme, un mot de passe pourrait suffir. Par contre si la somme est très importante, votre banquier va procéder à une authentification forte (Ex: vous poser plusieurs questions (défis) pour vous authentifier).

Authentification faible

Vulnérabilité d'authentification d'entité faible par mot de passe fixe :

- possibilité d'intercepter le mot de passe
- mots de passes stockés dans un fichier protégé en lecture et écriture
- attaque par dictionnaire
- etc..

Authentification forte avec mots de passe à usage unique

Protège contre les interceptions de message

Basée sur une liste pré-partagée

Basée sur un incrément

Basée sur une fonction à sens unique : exemple protocole de Lamport (voir diapositive suivante)

Authentification forte avec mots de passe à usage unique : exemple protocole de Lamport

- A veut s'identifier à B
- A possède un secret w,
- h fonction à sens unique connue par A et B
- t : constante définissant le nombre d'authentifications autorisées (exemple: si t=100, après 100 authentification, A génère un nouveau secret w)
- On définit par récurrence : $h^0(s)=s$, $h^i(s)=h^{i-1}(h(s))$ pour $1<=i<=t$
- On note $wi=h^{t-i}(s)$, $i^{-ième}$ mot de passe
- A transfère à B par un canal sûr $w_0=h^t(s)$,
- B initialise son compteur $compt_a$ à 1
- À la $i^{ème}$ identification :
    - A calcule w_i, et envoie à B M1 : A->B : $a.i.w_i$
    - B vérifie que $compt_a=i$ et que $h(w_i)=w_{i-1}$

Authentification forte par défi réponse

Une entité (le prouveur) prouve son identité à une autre entité (le vérificateur) en démontrant au vérificateur qu'il possède un secret sans révéler ce secret grâce à une réponse à un *challenge* variant dans le temps.

Le *challenge (défi)* peut être :

- nonce : nombre pseudo aléatoire non prévisible par un attaquant
- numéro de séquence : nécessite de mémoriser les numéros de séquence déjà utilisés
- estampille (horodateur, time stamp) : nécessite des horloges synchronisées et sécurisées
- combinaison de nonce avec numero de sequence pour garantir qu’un nonce n’a pas été dupliquée

Authentification forte par défi réponse

Critères de classification :

- Systèmes cryptographiques à clés publiques ou symétriques
- Nombre de messages
- Authentification unilatérale ou mutuelle

Types de protocoles par défi-réponse

- Protocoles utilisant une clé secrète (chiffrement symétrique ou MAC)
- Protocoles basés sur un chiffrement avec clé publique
- Protocoles basés sur la vérification d'une signature

Authentification forte par défi-réponse basée sur une clé partagée (3 variantes)

Variante 1

- M1: a->b : I'm a
- M2: b->a : rb
- M3: a->b : {rb}Kab

Variante 2

- M1: a->b : I'm a
- M2: b->a : {rb}Kab
- M3: a->b : rb

Variante 3 : une passe

- M1: a->b : a.{ta}Kab

Nécessite que a et b aient des horloges synchronisées. b déchiffre le message et s'assure que ta est dans un intervalle de temps raisonnable

Inconvénient d'usage de clés partagées : Si la base de donnée du côté serveur (B) est corrompue, Alice peut être usurpée par un intrus. Solution: usage des clés publiques

Authentification forte par défi-réponse à base de clés publiques

Variante 1

- M1: a->b : I'm a
- M2: b->a : rb
- M3: a->b : {rb}SKa (a signe le nonce rb avec sa clé privée)

Variante 2

- M1: a->b : I'm a
- M2: b->a : {rb}PKa (b chiffre le nonce rb avec la clé publique de a)
- M3: a->b : rb

Attention : l'authentification est unilatérale : un attaquant peut faire signer à a un message, ou intercepter un message qui lui est destiné et le lui faire déchiffrer !

Authentification forte par défi réponse : Authentification mutuelle à base de clé partagée

Version 1

- M1: a->b : I'm a
- M2: b->a : rb
- M3: a->b : {rb}Kab
- M4: a->b : ra
- M5: b->a : {ra}Kab

Version 2 (avec 3 messages seulement)

- M1: a->b : I'm a . ra
- M2: b->a : rb.{ra}Kab
- M3: a->b : {rb}Kab

Attention les deux versions sont *vulnérables* aux attaques réflexives avec entrelacement de session (voir diapositive suivante)

Attaque réflexive avec entrelacement de sessions

Session 1	Session 2
- M1 : c/a -> b : l'm a . ra - M2 : b -> c/a : rb.{ra}Kab	c/a ne sait pas répondre au challenge **rb** donc il va ouvrir session 2 pour le demander
à la fin de la session 2; c/a a obtenu la réponse au challenge **rb** qui est **{rb}Kab**	- M1 : c/a -> b : l'm a . **rb** - M2 : b -> c/a : rb'.**{rb}Kab**
- M3 : c/a -> b : **{rb}Kab**	c/a a répondu au challenge **rb** avec **{rb}Kab**

Solution

- M1: a->b : I'm a . ra
- M2: b->a : rb.{ra.**b**}Kab
- M3: a->b : {rb.**a**}Kab

Authentification forte avec l’utilisant des estampilles à la place de nonces

Ca permet de réduire le nombre de messages mais nécessite la synchronisation des horloges de a et b.

- M1: a->b : I'm a . {a.ta}Kab
- M2 : b->a : I’m b . {b.ta}Kab

Authentification forte par défi réponse : Authentification mutuelle à base de clé publique

Version 1

- M1: a->b : I'm a . {ra}PKb
- M2: b->a : ra.{rb}PKa
- M3: a->b : rb

Version 2

- M1: a->b : I'm a . ra
- M2: b->a : rb . {ra}SKb
- M3: a->b : {rb}Ska

Comment a connaît la clé publique de b ? Ces protocoles sont vulnérables aux attaques "man in the middle". Pour résoudre le problème, il faut introduire la certification des clés publiques par une autorité de confiance.

Références bibliographiques

Yacine Challal, *Ingénierie des protocoles et logiciels sécurisés*
http://y_challal.esi.dz/index.php/2016/04/30/ingenierie-des-protocoles-et-logiciels-securises-ipls/

James F. Kurose, Keith W. Ross Computer Networking: A Top-Down Approach, 6th Edition (2013)

https://www.pearson.com/us/higher-education/product/Kurose-Computer-Networking-A-Top-Down-Approach-6th-Edition/9780132856201.html