Définition : Protocole cryptographique

Un protocole cryptographique est une séquence d'étapes spécifiant les actions respectives devant être réalisée par deux entités (ou plus) pour remplir un objectif de sécurité donné (exemple: Diffie Hellman pour l'échange de clé).

Remarque : Protocole vs. algorithme cryptographique

Un protocole cryptographique utilise des algorithmes de cryptographie.

Typologie des protocoles cryptographiques

Les protocoles cryptographiques peuvent être classés en plusieurs catégories :

- Protocoles d'échange de clefs (key exchange, key agreement)
  - Création d'un secret partagé (exemple: 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)
  - Horodatage et estampillage
  - ...

Syntaxe : Notation

Nous allons adopter la notation suivante pour la spécification de protocoles cryptographiques :

- Kab désignera une clé secrète (utilisée dans un algorithme symétrique) partagée entre a et b
- PKa désignera une clé publique de a(utilisée dans un algorithme asymétrique)
- SKa désignera une clé privée de a (utilisée dans un algorithme asymétrique)
- Si m désigne un message
  - {m}Kab désignera m chiffré avec Kab
  - {m}PKa désignera m chiffré avec PKa
  - {m}SKa désignera m chiffré avec SKa
  - H(m) désignera un condensé calculé sur m avec la fonction de hachage h
  - H_k(m) désignera un MAC calculé sur m avec la fonction de hachage H paramétrée avec la clé k, H_k.
- Protocole := Message ... Message
- Message := M_n, Entité -> Entité : Data
- Entité := A|B|S|C/A|C/B|C/S (A: Alice, B: Bob, C:

Attaquant, S: Serveur) Data :=
A|B|S
K tel que K dans
{Ksa,Ksb,Kab,PKa,PKb,PKs,SKa,SKb,SKs}
ra|rb (Nonces générés respectivement
par a et b)
ta|tb (estampilles générées
respectivement par a et b) Data.Data
(concaténation)
{Data}k donnée
chiffrée avec k
Data* (donnée
optionnelle)

- Nonce (oNly ONCE) (ou aléas) : nombre unique et imprévisible
- Estampille : marqueur de temps, sert à calculer la fraîcheur

Hypothèses sur l'attaquant

- 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

B. Protocoles d'échange de clé

Echange de clé en utilisant un système asymétrique

M1: b->a : PKb

M2: a->b : {Kab}PKb

Attention : Attaque man in the middle

Ne garantie pas l'authentification: attaque « man in the middle »

M1: b->c/a: PKb

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

Protocoles d'échange de clé : Otway-Rees

M1: a->b: m.a.b.{na.m.a.b}Ksa (m identifiant de transaction)

M2: b->s: m.a.b.{na.m.a.b}Ksa.{nb.m.a.b}Ksb

M3: s->b: m.{na.Kab}Ksa.{nb.Kab}Ksb M4: b->a: m.{na.Kab}Ksa

Attention : Attaque par confusion de type

Supposons que m fait 32 bits, a et b 16 bits, et Kab 64 bits. Il suffit que C rejoue la partie chiffrée de M1

M1: a->c/b: m.a.b.{na.m.a.b}Ksa

M4': c/b->a: m.{na.m.a.b}Ksa

Conclusion a accepte m.a.b comme nouvelle clé (envoyée en claire !!!)

C. Protocoles d'authentification de l'origine

Objectif b doit être assuré que le message a été créé par a. On peut assurer l'authentification de l'origie de deux manières :

- Utilisation d'un algorithme asymétrique de signature

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

- Utilisation d'un MAC

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

D. Protocoles d'authentification d'entité

Objectif

A doit être authentifié auprès de b à la fin du déroulement du protocole

Types d'identification

- Authentification faible : mots de passe fixes, mots de passe à usage unique
- Authentification forte : protocoles défi-réponse basés sur la cryptographie symétrique ou asymétrique

Attention : 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, ou
- Utilisation de fonction à sens unique avant le stockage (exemple: login sous unix)
- Attaque par dictionnaire

E. Authentification forte par défi réponse

Principe

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

Définition : Challenge

Utilisation de nonce garantissant une notion de fraîcheur / unicité

- **Nombre pseudo-aléatoire (noté r dans la suite)**
- **Nombre non prévisible par un attaquant**
- **Numéro de séquence (noté n dans la suite)**
- **Nécessite de mémoriser les numéros de séquence déjà utilisés**

- **Estampille (horodateur, time stamp) + horloges synchronisées (noté t)**
- **Les horloges doivent être sécurisées**
- **Combinaison de nonces : nombre aléatoire concaténé à un numéro de séquence ou à une estampille**
- **Permet de garantir qu'un nombre aléatoire n'a pas été dupliqué**

Remarque : Critères de classification

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

Remarque : 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**

1. Authentification forte par défi-réponse basée sur une clé partagée

Variante 1

- **M1: a->b : I'm A**
- **M2: b->a : nb**
- **M3: a->b : {nb}Kab**

L'authentification est non mutuelle

Variante 2

- **M1: a->b : I'm A**
- **M2: b->a : {nb}Kab**
- **M3: a->b : nb**

Variante 3 : une passe

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

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

Remarque : 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

2. **Authentification forte par défi-réponse à base de clés publiques**

Variante 1

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

Variante 2

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

Attention : Limitation---------- expl

L'authentification est unilatérale => un intrus peut faire signer à A un message, ou intercepter un message qui lui est destiné et le lui faire déchiffrer !!!

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

Version 1

- **M1: a->b : I'm A**
- **M2: b->a : nb**
- **M3: a->b : {nb}Kab**
- **M4: a->b : na**
- **M5: b->a : {na}Kab**

Version 2 : réduire le nombre de messages

- **M1: a->b : I'm A . na**
- **M2: b->a : nb.{na}Kab**
- **M3: a->b : {nb}Kab**

Attention : Vulnérabilité : Attaque réflexive avec entrelacement de sessions

Session 1	Session 2
M1: c/a->b : I'm A . na M2: b->c/a : nb.{na}Kab	
	M1: c/a->b : I'm A . nb M2: b->c/a : nb'.{nb}Kab
M3: c/a->b : {nb}Kab	

Version 3 : Réduire le nombre de messages en utilisant des estampilles à la place de nonces
Nécessite la synchronisation des horloges de A et B

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

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

Version 1

- **M1: a->b : I'm A . {na}PKb**
- **M2: b->a : na.{nb}PKa**
- **M3: a->b : nb**

Version 2

- **M1: a->b : I'm A . na**
- **M2: b->a : nb . {na}SKb**
- **M3: a->b : {nb}Ska**

Remarque

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.