

# Réseaux Sécurité

E. Jeandel

Emmanuel.Jeandel at lif.univ-mrs.fr

- 1 Généralités
- 2 Cryptographie
- 3 Protocoles
- 4 Sécurité dans les réseaux

- Communiquer de façon sûre, sans être écouté ni attaqué

# Ennemis

## Enemy at the Gates

- renifleur (sniffer) de paquets
  - Interface réseau en mode *promiscuous*
- Espionne les *trames*
- couche liaison
- ARP Poisoning

# Ennemis

## Behind Enemy Lines

- Usurpation d'adresse IP (IP spoofing)
- Permet a priori juste d'*envoyer* des paquets, mais pas d'en recevoir..
  - ..sauf si on arrive à changer les tables de routage

# Ennemis

## Public Enemy

- Déni de Service (DoS, ou *Denial of Service*)
- Empêcher une machine ou une infrastructure de faire quoi que ce soit en la bombardant de requêtes à traiter
- Utilise souvent l'IP spoofing
- SYN flood
  - Pour chaque SYN, le serveur alloue de la mémoire..
- smurf DoS
  - *E* se fait passer pour *A* et envoie des paquets ICMP ping en broadcast à plein de machines.

3 objectifs :

- *Disponibilité* : Les données sont utilisables quand on les demande
- *Intégrité* : Les données ne peuvent pas être altérées
- *Confidentialité* : Les données ne sont connues que des personnes ayant besoin de les connaître

- Chiffrement
- Identification
- Authentification
- Signatures
- Pares-Feu

- 1 Généralités
- 2 **Cryptographie**
- 3 Protocoles
- 4 Sécurité dans les réseaux

## Plusieurs mécanismes

- Cryptographie symétrique
- Cryptographie à paire (clé publique, clé privée)
- Fonctions unidirectionnelles

## Hypothèses :

- Les protocoles sont publics
- Possibilité d'avoir un tiers de confiance

- On a deux fonctions  $D(k, x)$  et  $E(k, x)$  qui permettent respectivement de coder et de décoder le message  $x$  avec la clé  $k$ .
- $D(k, (E(k, x))) = x$
- On notera plutôt  $D_k$  et  $E_k$ .
- Exemple : DES, Rijndael (AES).

Propriétés voulues :

- Il est “facile” de calculer  $D_k(x)$  (resp  $E_k(x)$ ) connaissant  $k$  et  $x$ .
- Il est “difficile” de calculer  $x$  connaissant  $E_k(x)$ .
- Il est “difficile” de générer un  $y$  tel que  $E_k(x) = y$ .
- Il est “difficile” de calculer  $k$  connaissant  $E_k(x)$  et  $x$ .
- La sécurité de l’algorithme réside dans la clé  $k$ .

En général,

- “facile” : en temps polynomial
- “difficile” : conjecturé impossible en temps polynomial sauf si **P = NP**

# Cryptographie à clé publique

- Deux clés  $k_1$  et  $k_2$ .  $k_1$  est connue de tout le monde (clé publique),  $k_2$  est secrète (clé privée).
- $D_{k_2}(E_{k_1}(x)) = x$
- Tout le monde peut encrypter, seul ceux connaissant  $k_2$  peuvent décrypter.
- Exemple : RSA.

Variantes :

- $D_{k_1}(E_{k_2}(x)) = x$
- Tout le monde peut décrypter, seul ceux connaissant  $k_2$  peuvent encrypter.

Autre variante : les deux à la fois.

Propriétés voulues :

- Il est “facile” d’encrypter (de calculer  $E_{k_1}(x)$ ) connaissant  $k_1$  et  $x$ .
- Il est “difficile” de trouver  $x$  connaissant  $E_{k_1}(x)$ .
- Il est “difficile” de trouver  $k_2$  connaissant  $k_1$ .
- La sécurité de l’algorithme réside dans la clé privée  $k_2$ .

# Fonctions unidirectionnelles (de hachage)

- Une fonction  $f$  qui envoie tout mot sur un mot de  $n$  bits (pour un certain  $n$ ).
- Exemple : MD5, SHA.

# Fonctions unidirectionnelles

- Il est facile de calculer  $f(x)$  connaissant  $x$
- Il est difficile de trouver un  $y$  tel que  $f(y) = x$ .
- Il est difficile de trouver  $x, y$  tel que  $f(y) = f(x)$ .

## Exercice

- Combien de chaînes  $y$  faut-il tester avant d'en trouver une (avec probabilité 0.5) telle que  $f(y) = x$  ?
- Combien de chaînes faut-il tester avant d'en trouver deux  $x, y$  telles que  $f(y) = f(x)$  ?

- 1 Généralités
- 2 Cryptographie
- 3 Protocoles**
- 4 Sécurité dans les réseaux

- 1 Généralités
- 2 Cryptographie
- 3 Protocoles
  - Authentification
  - Signatures
  - Clé partagée
  - Intermédiaires
- 4 Sécurité dans les réseaux
  - Couche physique
  - Couche liaison
  - Couche Réseau (IP)
  - Couche transport
  - Couche Sécurité

- Alice discute sur un canal avec Bob
- Alice veut prouver à Bob qu'il s'agit bien d'elle

Exemple :

- Bob = Banque, Alice = Client
- Alice = Client, Bob = Banque

Un étudiant malintentionné peut écouter les communications.  
Exemple : authentification par mot de passe.

- Ne pas transmettre le mdp en clair
- Ne pas transmettre le mdp crypté
  - Attaque replay/playback

# Solution 1

## Clé symétrique

Avec une clé symétrique, partagée par Alice et Bob

- Alice déclare à Bob : “Je suis Alice”
- Bob envoie un *nonce*  $x$  (number used once)
- Alice crypte  $x$
- Bob vérifie
- *Challenge-Response* authentication
- Principe similaire : mdp différent à chaque connexion (*one time password*)

# Solution 2

## Clé publique

Bob connaît la clé publique d'Alice.

- Alice dit “je suis Alice”
- Bob choisit un *nonce*  $x$
- Alice envoie  $k$  à Bob encrypté :  $E_{k_1}(k)$
- Bob décrypte la réponse d'Alice

Problème :

- Comment Bob obtient-il la clé d'Alice ?
- Si c'est Alice qui l'envoie, faille dans le protocole
- *Attack man-in-the-middle*

## 1 Généralités

## 2 Cryptographie

## 3 Protocoles

- Authentification
- Signatures
- Clé partagée
- Intermédiaires

## 4 Sécurité dans les réseaux

- Couche physique
- Couche liaison
- Couche Réseau (IP)
- Couche transport
- Couche Sécurité

Alice veut signer un document électronique de sorte que

- Personne ne peut signer à sa place sans qu'on s'en rende compte.
- On se rend compte qu'elle a signé
- La signature n'est pas déplaçable ou supprimable.
- On ne peut pas modifier le document après signature

Soit  $x$  le document.

- Alice calcule  $y = E_{k_2}(x)$ .
- Le résultat est le document  $y$ .

Caractéristiques ?

- Personne ne peut signer à sa place sans qu'on s'en rende compte ?
- On se rend compte qu'elle a signé ?
- La signature n'est pas déplaçable ou supprimable ?
- On ne peut pas modifier le document après signature ?

Soit  $x$  le document.

- Alice calcule  $y = E_{k_2}(f(x))$
- Le résultat est le document  $x$ , adossé à  $y$

Ajout important : une date de validité du document (pour éviter de l'utiliser plusieurs fois)

## 1 Généralités

## 2 Cryptographie

## 3 Protocoles

- Authentification
- Signatures
- Clé partagée
- Intermédiaires

## 4 Sécurité dans les réseaux

- Couche physique
- Couche liaison
- Couche Réseau (IP)
- Couche transport
- Couche Sécurité

- Obtenir une clé  $k$  commune à Alice et Bob
- Plus loin, solution avec tiers de confiance

- Suppose qu'il est difficile, connaissant  $g^x \pmod p$  et  $g$  de retrouver  $x$

Principe :

- Alice choisit  $a$ , envoie  $g^a \pmod p$  à Bob
- Bob choisit  $b$ , envoie  $g^b \pmod p$  à Alice
- La clé est alors  $g^{ab} \pmod p$

## 1 Généralités

## 2 Cryptographie

## 3 Protocoles

- Authentification
- Signatures
- Clé partagée
- Intermédiaires

## 4 Sécurité dans les réseaux

- Couche physique
- Couche liaison
- Couche Réseau (IP)
- Couche transport
- Couche Sécurité

- Tiers de confiance

Résoudre deux problèmes :

- Permettre à Alice et Bob de s'entendre sur une clé symétrique
- Permettre à Alice d'obtenir la clé publique de Bob

- *Key Distribution Center*
- A une clé symétrique avec chacun des participants
- Quand Alice veut parler à Bob :
  - Alice demande au KDC
  - Le KDC choisit une clé  $k$
  - Le KDC envoie à Alice
    - La clé  $k$
    - $(Alice, k)$  encrypté avec la clé de Bob

- *Kerberos*
- KDC = “Kerberos Authentication Server” (AS) + Ticket-Granting Server (TGS)
- La paire (*Alice, k*) dans ce contexte s’appelle un *ticket*
- Plus de précisions dans un TD d’[Outils de l’Internet](#)

- *Certification Authority*
- Un CA *certifie* (signe) la clés publique d'Alice
- Si on a confiance au CA, on a confiance en la clé publique
- Type de certificat défini dans ITU X.509 et RFC 1422
- Approche différente : *Web of Trust* (PGP, etc) sans CA

**Ce certificat a été vérifié pour les utilisations suivantes :**

Certificat serveur SSL

**Émis pour**

Nom commun (CN)	casu1u3.phocean.fr
Organisation (O)	Université Aix-Marseille 1 Provence
Unité d'organisation (OU)	<Ne fait pas partie du certificat>
Numéro de série	00:81:B7:86:E1:71:5A:1B:77:CF:06:DD:D7:11:EA:0D:26

**Émis par**

Nom commun (CN)	TERENA SSL CA
Organisation (O)	TERENA
Unité d'organisation (OU)	<Ne fait pas partie du certificat>

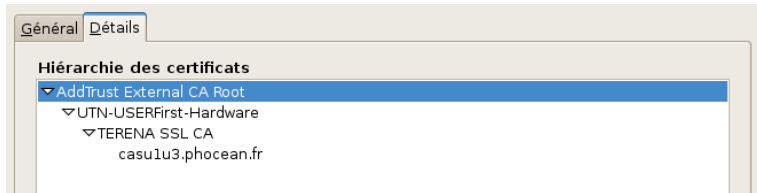
**Validité**

Émis le	18/12/2009
Expire le	18/12/2012

**Empreintes numériques**

Empreinte numérique SHA1	28:C6:21:91:8D:6C:8A:53:17:18:D9:6D:50:D5:51:4A:AB:9D:66:29
Empreinte numérique MD5	90:33:72:EF:A4:91:1B:F3:E1:C1:F7:1B:D5:DC:07:EF

- Public Key Infrastructure
- Sous forme d'arbre
  - L'application fait confiance à certains CA
  - Si on reçoit un certificat émis par un CA  $x$  inconnu, et certifié par un CA  $y$ , on essaiera de certifier  $y$
  - Chaque application doit avoir une liste des CA "racines"
  - En général un certificat contiendra l'ensemble des certificats des CA jusqu'à la racine
  - Exemple : `ls -l /etc/ssl/certs`



```
monster$ ls /usr/share/ca-certificates/mozilla/  
AddTrust_External_Root.crt  
AddTrust_Low-Value_Services_Root.crt  
AddTrust_Public_Services_Root.crt  
AddTrust_Qualified_Certificates_Root.crt  
..  
UTN_DATACorp_SGC_Root_CA.crt  
UTN_USERFirst_Email_Root_CA.crt  
UTN_USERFirst_Hardware_Root_CA.crt  
UTN-USER_First-Network_Applications.crt  
..
```

- 1 Généralités
- 2 Cryptographie
- 3 Protocoles
- 4 Sécurité dans les réseaux**

## 1 Généralités

## 2 Cryptographie

## 3 Protocoles

- Authentification
- Signatures
- Clé partagée
- Intermédiaires

## 4 Sécurité dans les réseaux

- Couche physique
- Couche liaison
- Couche Réseau (IP)
- Couche transport
- Couche Sécurité

# Couche physique



## 1 Généralités

## 2 Cryptographie

## 3 Protocoles

- Authentification
- Signatures
- Clé partagée
- Intermédiaires

## 4 Sécurité dans les réseaux

- Couche physique
- Couche liaison
- Couche Réseau (IP)
- Couche transport
- Couche Sécurité

- Réseau câblé
  - Aucun protocole existant
  - Solution pour l'ARP spoofing :
    - Tables ARP statiques sur les switches
    - *Port Security* : Un port physique = une MAC
- Sans-fil
  - Wired Equivalent Privacy (WEP)
  - Wi-Fi Protected Access (WPA, WPA2)

## 1 Généralités

## 2 Cryptographie

## 3 Protocoles

- Authentification
- Signatures
- Clé partagée
- Intermédiaires

## 4 Sécurité dans les réseaux

- Couche physique
- Couche liaison
- Couche Réseau (IP)
- Couche transport
- Couche Sécurité

# Couche réseau

- IPsec
- Un champ dans le paquet permet de spécifier le protocole cryptographique utilisé
- Protocole *connecté* (*Security Agreement*) *unidirectionnel*
- Utilise en général Diffie-Hellmann pour donner la clé

## Deux modes

- Transport
- Tunnel

## Deux protocoles principaux :

- Authentication Header (AH) protocol (Authentification, Intégrité, pas de crypto)
- Encapsulation Security Payload (Tout)

- Seules les données du paquets sont cryptées
- Peut être routé
- Problème : ne passe pas un NAT

- Le paquet est entièrement crypté puis inséré dans un autre paquet IP
- VPN
- Si on l'insère dans un paquet *UDP* plutôt qu'*IP*, permet de passer un NAT

## 1 Généralités

## 2 Cryptographie

## 3 Protocoles

- Authentification
- Signatures
- Clé partagée
- Intermédiaires

## 4 Sécurité dans les réseaux

- Couche physique
- Couche liaison
- Couche Réseau (IP)
- Couche transport
- Couche Sécurité

- Travaille au niveau transport et réseau
- Filtrage d'adresses IP et de ports
- Difficultés
  - Suivre les connexion TCPs
  - Protocoles à port variable (FTP)
- Exemple : filtrer tous les paquets TCP SYN (sans ACK) entrants
  - Que se passe-t-il ?

## 1 Généralités

## 2 Cryptographie

## 3 Protocoles

- Authentification
- Signatures
- Clé partagée
- Intermédiaires

## 4 Sécurité dans les réseaux

- Couche physique
- Couche liaison
- Couche Réseau (IP)
- Couche transport
- Couche Sécurité

- TLS (Transport Layer Security)
- Entre la couche Transport et les couches applicatives
- Successeur de SSL (Secure Sockets Layer)
- Relativement compliqué

- Créer d'abord la connection TCP

```
SSL_CTX *SSL_CTX_new(const SSL_METHOD *method);  
SSL *SSL_new(SSL_CTX *ctx);  
int SSL_set_fd(SSL *ssl, int fd);  
int SSL_connect(SSL *ssl);  
int SSL_read(SSL *ssl, void *buf, int num);  
int SSL_write(SSL *ssl, const void *buf, int num);
```

- method : version du protocole supporté