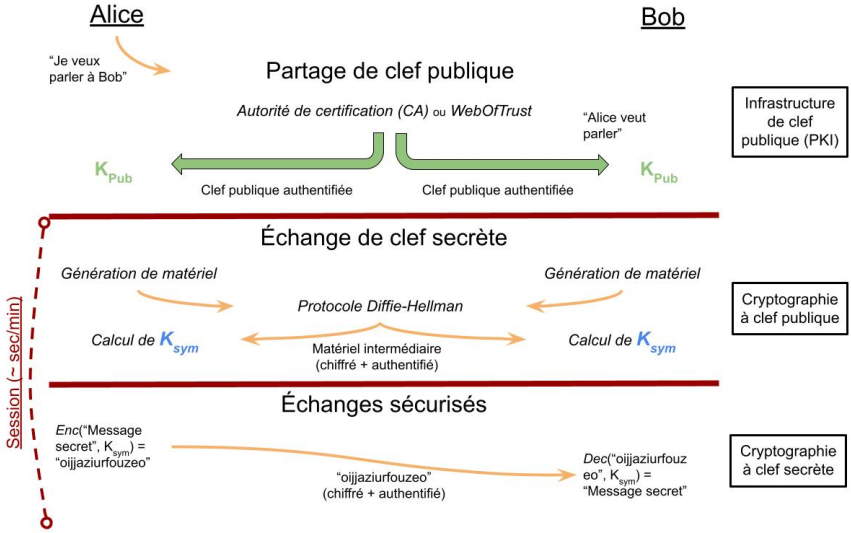
Ch. 7 PKI : Infrastructures de clefs publiques (Distribution et échange de clefs)

UE Master 2 : Sécurité des Systèmes d'Information

Sébastien Duval
sebastien.duval@loria.fr

Contenu

Communication classique sécurisée (TLS)



Distribution et échange

Distribution

Si Bob veut recevoir des messages, pas besoin de répondre, pas besoin d'échange de clefs mais juste de publier/distribuer sa propre clef publique.

Échange

S'il faut une communication bilatérale, il faut un échange : ça ne fonctionne pas pareil, même si on peut faire une distribution dans chaque sens, on peut faire plus efficace.

Clef publique ou clef secrète ?

D'après vous ?

Est-ce que parfois on peut se passer de toute la partie clef publique ?

Clef publique ou clef secrète ?

D'après vous ?

Est-ce que parfois on peut se passer de toute la partie clef publique ?

Oui !

Clef publique ou clef secrète ?

La cryptographie **à clef publique** sert à *créer* un canal sécurisé entre 2 entités *qui ne se connaissent pas* !

Si les 2 entités se connaissent, elles peuvent déjà partager un secret à l'avance !
e.g. : réseau interne d'une entreprise, connexion à distance à votre propre machine (SSH, Remote-Desktop ou autres), ...

Clef publique ou clef secrète ?

La cryptographie **à clef publique** sert à *créer* un canal sécurisé entre 2 entités *qui ne se connaissent pas* !

Si les 2 entités se connaissent, elles peuvent déjà partager un secret à l'avance !
e.g. : réseau interne d'une entreprise, connexion à distance à votre propre machine (SSH, Remote-Desktop ou autres), ...

Et en pratique ?

- les protocoles à clef publique sont plus automatisés
  $\Rightarrow$ ils sont utilisés même quand de la clef secrète suffirait !
  e.g. SSH, beaucoup de réseaux internes d'entreprises, ...
- une exception majeure : **Kerberos**
  e.g. quelques réseaux internes et presque tous les Active Directory

Les nonces dans les protocoles

Nonce (définition en cryptographie)

Un **nonce** est une valeur publique *aléatoire* qui ne doit être utilisée qu'*une fois*.

Les nonces dans les protocoles

Nonce (définition en protocoles)

Un **nonce** est une valeur ~~publique~~ *aléatoire* qui ne doit être utilisée qu'*une fois*.

En protocoles, le nonce est généralement considéré comme secret (c'est un abus de terminologie mais c'est comme ça).

Usage du nonce dans les protocoles

- essentiel
- sert de **challenge d'authentification** :

a. généralement rechiffré avec une autre clef, celle d'Alice, pour éviter que le retour ne soit attaqué

Les nonces dans les protocoles

Nonce (définition en protocoles)

Un **nonce** est une valeur ~~publique~~ *aléatoire* qui ne doit être utilisée qu'*une fois*.

En protocoles, le nonce est généralement considéré comme secret (c'est un abus de terminologie mais c'est comme ça).

Usage du nonce dans les protocoles

- essentiel
- sert de **challenge d'authentification** :
  - Alice envoie un nonce chiffré à Bob

a. généralement rechiffré avec une autre clef, celle d'Alice, pour éviter que le retour ne soit attaqué

Les nonces dans les protocoles

Nonce (définition en protocoles)

Un **nonce** est une valeur ~~publique~~ *aléatoire* qui ne doit être utilisée qu'*une fois*.

En protocoles, le nonce est généralement considéré comme secret (c'est un abus de terminologie mais c'est comme ça).

Usage du nonce dans les protocoles

- essentiel
- sert de **challenge d'authentification** :
  - Alice envoie un nonce chiffré à Bob
  - Est-ce que Bob arrive à déchiffrer ? Si oui, c'est qu'il a la clef secrète : ça prouve son identité.

---

a. généralement rechiffré avec une autre clef, celle d'Alice, pour éviter que le retour ne soit attaqué

Les nonces dans les protocoles

Nonce (définition en protocoles)

Un **nonce** est une valeur ~~publique~~ *aléatoire* qui ne doit être utilisée qu'*une fois*.

En protocoles, le nonce est généralement considéré comme secret (c'est un abus de terminologie mais c'est comme ça).

Usage du nonce dans les protocoles

- essentiel
- sert de **challenge d'authentification** :
  - Alice envoie un nonce chiffré à Bob
  - Est-ce que Bob arrive à déchiffrer ? Si oui, c'est qu'il a la clef secrète : ça prouve son identité.
  - Alice attend de recevoir le nonce déchiffré [a]

a. généralement rechiffré avec une autre clef, celle d'Alice, pour éviter que le retour ne soit attaqué

Première partie I

Distribution de clefs

Contenu Partie I : Distribution de clefs

Distribution/Génération de clef secrète

Comment générer et partager une nouvelle clef secrète ?

- à partir d'une ancienne clef
  $\Rightarrow$ mais il faut une clef de départ
- tiers de confiance
  $\Rightarrow$ similaire aux CA, on les appelle KDC (Key-Distribution Center)

Remarque : une option pour avoir une clef de départ est de faire un premier partage de clef en cryptographie à clef publique.
Une autre option est d'insérer une clef secrète dans la machine à sa fabrication/mise en production.

Protocole basique de distribution par KDC

1. $A \rightarrow KDC$: requête, N_1
2. $KDC \rightarrow A$: $\{K_s, \text{requête}, N_1\}_{K_a}$, $\{K_s, ID_a\}_{K_b}$
3. $A \rightarrow B$: $\{K_s, ID_a\}_{K_b}$
4. $B \rightarrow A$: $\{N_2\}_{K_s}$
5. $A \rightarrow B$: $\{f(N_2)\}_{K_s}$

- 1–3 : distribution
- 3–5 : authentification

Note : KDC connaît la clef secrète.
Pas un problème si KDC est l'admin système qui gère la distribution des clefs des machines d'un parc informatique, par exemple.

Contenu Partie I : Distribution de clefs

Options pour distribuer une clef publique

- annonce publique
  - *e.g.* apposer la clef PGP aux mails, la poster sur un forum, etc
  - vulnérable aux usurpations d'identité
- annuaire d'identités public
  - il faut faire confiance à l'annuaire
  - vulnérable aux usurpations d'identité
- autorité de clef publique
  - annuaire de clefs publiques (*e.g.* DANE, cf. TD)
  - demander à l'autorité la clef publique du correspondant

  $\Rightarrow$ nécessite un accès temps réel
- **certificat de clef publique**
  - pas besoin d'accès temps réel
  - lie une identité à une clef publique
  - **signé** par la clef *privée* d'une entité de confiance (CA)
  - **vérifiable** par toute personne connaissant la clef *publique* de *la CA*

Distribution/Échange de clef publique

Les solutions de *distribution* de clef publique permettent toutes l'*échange* de clef publique.

On détaillera donc les solutions dans la partie sur l'échange de clefs.

Contenu Partie I : Distribution de clefs

Clefs de session : motivation

Contraintes de la distribution de clef publique :

- sous hypothèse d'une entité de confiance
- simple
- **lent**

Solution : système hybride

- échange d'un secret initial par asymétrique (clef publique)
- communication symétrique grâce à ce secret partagé
- quand il faut changer de clef symétrique, partager $\{K_{next}\}_{K_{current}}$

Problème : **"forward-secrecy"**
$\rightarrow$ si une clef est compromise, toutes les suivantes aussi

Clefs de session : motivation

Contraintes de la distribution de clef publique :

- sous hypothèse d'une entité de confiance
- simple
- **lent**

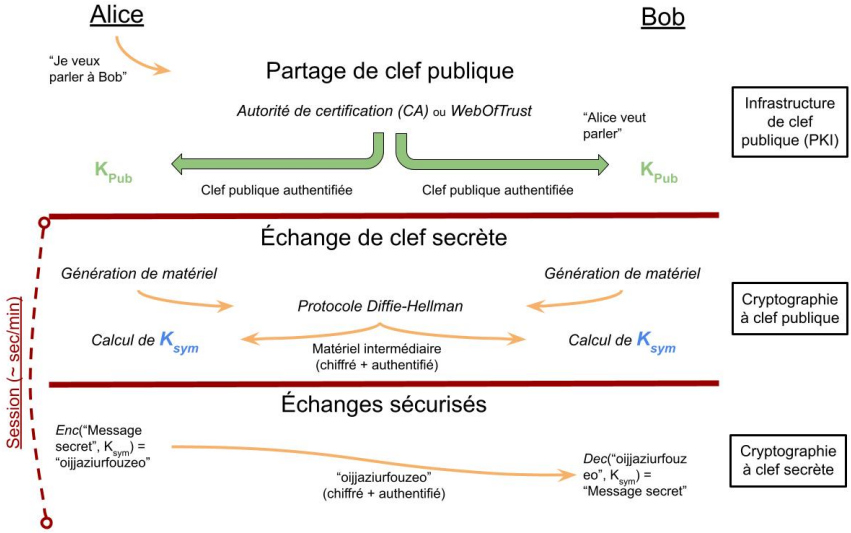
Solution : système hybride

- échange d'un secret initial par asymétrique (clef publique)
- communication symétrique grâce à ce secret partagé
- quand il faut changer de clef symétrique, partager $\{K_{next}\}_{K_{current}}$

Problème : **"forward-secrecy"**
$\rightarrow$ si une clef est compromise, toutes les suivantes aussi

Solution : **sessions**

Communication classique sécurisée (TLS)



Clefs de session : principe

Session

- système hybride
- changement de clef
  - indépendante de $K_{current}$
  - réinitialisation complète

  $\rightarrow$ on refait un échange de clef asymétrique (Diffie-Hellman en général)

  $\Rightarrow$ forward-secrecy :
  si une clef est compromise, vulnérabilité pendant la session courante
- sessions courtes ($\sim$ 5 minutes) [a]

a. quand on fait bien les choses

Distribution simple de clef de session

Merkle, 1979

1. A produit une paire (PK_a, SK_a) provisoire
2. A envoie à B sa clef publique et son identité
3. B produit la clef de session K_s et l'envoie à A, chiffré sous PK_a
4. A déchiffre K_s, on communique avec K_s

Attaque de la distribution simple de clef de session

Attaque

- $A \rightarrow B : PK_a, ID_a$
- $B \rightarrow A : \{K_s\}_{PK_a}$

Légende :
$I(B)$: imposteur qui se fait passer pour B
$I(A)$: imposteur qui se fait passer pour A
$\{x\}_K$: x chiffré avec la clef K

Faire un dessin : 2 personnes qui se serrent la main à travers un mur, mais en fait il y a un espace dans le mur et quelqu'un qui serre la main des 2 côtés (note : repérable, il n'a pas 2 mains droites !)

Attaque de la distribution simple de clef de session

Attaque

1. $A \rightarrow B : PK_a, ID_a$
2. $B \rightarrow A : \{K_s\}_{PK_a}$

Mais ! attaque "man-in-the-middle"

Légende :
$I(B)$: imposteur qui se fait passer pour B
$I(A)$: imposteur qui se fait passer pour A
$\{x\}_K$: x chiffré avec la clef K

Faire un dessin : 2 personnes qui se serrent la main à travers un mur, mais en fait il y a un espace dans le mur et quelqu'un qui serre la main des 2 côtés (note : repérable, il n'a pas 2 mains droites !)

Attaque de la distribution simple de clef de session

Attaque

1. $A \rightarrow B : PK_a, ID_a$
2. $B \rightarrow A : \{K_s\}_{PK_a}$

Mais ! attaque "man-in-the-middle"

1. $A \rightarrow I(B) : PK_a, ID_a$
2. $I(A) \rightarrow B : PK_i, ID_a$
3. $B \rightarrow I(A) : \{K_s\}_{PK_i}$
4. $I(B) \rightarrow A : \{K_s\}_{PK_a}$
5. $A \rightarrow I(B) : \{M\}_{K_s}$

Légende :
$I(B)$: imposteur qui se fait passer pour B
$I(A)$: imposteur qui se fait passer pour A
$\{x\}_K$: x chiffré avec la clef K

Faire un dessin : 2 personnes qui se serrent la main à travers un mur, mais en fait il y a un espace dans le mur et quelqu'un qui serre la main des 2 côtés (note : repérable, il n'a pas 2 mains droites !)

Scénario de distribution de clef de session avec confidentialité et authentification

Distribution sécurisée

1. $A \rightarrow B : \{N_1, PK_a, ID_a\}_{PK_b}$
2. $B \rightarrow A : \{N_1, N_2\}_{PK_a}$
3. $A \rightarrow B : \{N_2\}_{PK_b}$
4. $A \rightarrow B : \{K_s\}_{PK_b}$

Légende :
$\{x\}_K$: x chiffré avec la clef K
N : nonce (rappel : valeur aléatoire à ne jamais utiliser 2 fois avec la même clef)

Note : Il faut que A connaisse PK_b au départ. C'est le cas ici : on a déjà un canal en cours, on est en train de renouveler la clef de session. PK_a et K_s sont neuves mais pas PK_b.

Deuxième partie II

Échange de clefs

Contenu Partie II : Échange de clefs

Authentification : types

Objectif : authentification **mutuelle**

- protocoles dissymétriques
- authentification indirecte (via une clef de session)

Types d'authentification :

- **authentification et intégrité de clef** (*i.e.* vérifier que la clef n'a pas été modifiée et appartient bien à celui qui la donne)
- **authentification d'identité** (*i.e.* vérifier l'identité de notre interlocuteur)

Quelles clefs échanger ?

On veut échanger 3 types de clefs :

- **clefs (secrètes) de session** (*e.g.* TLS), grâce à un protocole à clef publique (ECDHE),
- **clefs secrètes directement** (*e.g.* réseau interne d'entreprise), grâce à un protocole à clef secrète (Kerberos),
- **clefs publiques** (*e.g.* pour initialiser TLS), grâce à une PKI (Infrastructure de Clefs Publiques).

Contenu Partie II : Échange de clefs

Principe de l'échange de clef de session (*e.g.* TLS)

Rappel : session pour "forward-secrecy"

- Objectif : partager une clef de session symétrique K_{ab} pour A et B qui soit "fraîche" (*i.e.* récente et indépendante de la précédente)
- on utilise :
  - une **clef publique éphémère** PK_{tmp}
  - une fonction de **hachage** cryptographique h
  - un **nonce** N_a

1. $A \rightarrow B : PK_{tmp}, N_a, \{h(PK_{tmp}, N_a)\}_{SK_a}$
2. $B \rightarrow A : \{K_{ab}\}_{PK_{tmp}}, \{h(K_{ab}, A, N_a)\}_{SK_b}$

Exemple : Diffie-Hellman

Diffie-Hellman, 1976

- échange de clef secrète ou clef de session
- utilisé partout (SSL/TLS)
- la valeur de la clef dépend des participants
- problème difficile : logarithme discret

1. A crée p premier et a primitif (publics)
2. A et B créent respectivement x_A et x_B
3. $A \rightarrow B$: $a^{x_A} \bmod p$
4. $B \rightarrow A$: $a^{x_B} \bmod p$
5. leur clef commune est $a^{x_A x_B} \bmod p$

ECDHE

En pratique, on n'utilise plus Diffie-Hellman.

On utilise **ECDHE** : Elliptic-Curve Diffie-Hellman Ephemeral.

- les a^{x_A} mod p deviennent des $a \times p$, où p est un point sur une courbe elliptique. C'est pareil, juste plus rapide et (a priori) légèrement plus sûr.
- **Ephemeral**, c'est un peu compliqué, mais là c'est un peu important à comprendre. C'est l'utilisation d'une clef éphémère dont on parlait il y a 2 slides.
- dans TLS 1.2, tout le monde utilisait ECDH ou ECDHE, même s'il y avait d'autres standards.
- depuis TLS 1.3, seul ECDHE est supporté.

Ephemeral

ECDH**E** dans TLS

- le développeur du serveur Web génère une paire (K_{pub}, K_{priv}),

Ephemeral

ECDH**E** dans TLS

1. le développeur du serveur Web génère une paire (K_{pub}, K_{priv}),
2. il l'ajoute dans la configuration TLS de son serveur et demande un certificat de la paire (URL, K_{pub}) à une autorité de certification,
   C'est tout côté développeur, le reste est transparent.

Ephemeral

ECDH**E** dans TLS

- le développeur du serveur Web génère une paire (K_{pub}, K_{priv}),
- il l'ajoute dans la configuration TLS de son serveur et demande un certificat de la paire (URL, K_{pub}) à une autorité de certification,
  C'est tout côté développeur, le reste est transparent.
- (K_{pub}, K_{priv}) est sauvegardée comme clefs-maîtresses du serveur,

Ephemeral

ECDH**E** dans TLS

1. le développeur du serveur Web génère une paire (K_{pub}, K_{priv}),
2. il l'ajoute dans la configuration TLS de son serveur et demande un certificat de la paire (URL, K_{pub}) à une autorité de certification,
   C'est tout côté développeur, le reste est transparent.
3. (K_{pub}, K_{priv}) est sauvegardée comme clefs-maîtresses du serveur,
4. à chaque demande de connexion d'un client, TLS génère une paire de clefs éphémères (K^{eph}_{pub}, K^{eph}_{priv}) adaptées pour des opérations sur les courbes elliptiques (pour ECDH),

Ephemeral

ECDH**E** dans TLS

1. le développeur du serveur Web génère une paire (K_{pub}, K_{priv}),
2. il l'ajoute dans la configuration TLS de son serveur et demande un certificat de la paire (URL, K_{pub}) à une autorité de certification,
   C'est tout côté développeur, le reste est transparent.
3. (K_{pub}, K_{priv}) est sauvegardée comme clefs-maîtresses du serveur,
4. à chaque demande de connexion d'un client, TLS génère une paire de clefs éphémères (K_{pub}^{eph}, K_{priv}^{eph}) adaptées pour des opérations sur les courbes elliptiques (pour ECDH),
5. une nouvelle clef de session symétrique K_{ab} est générée/partagée (*cf.* Slide 26),

Ephemeral

ECDH**E** dans TLS

1. le développeur du serveur Web génère une paire (K_{pub}, K_{priv}),
2. il l'ajoute dans la configuration TLS de son serveur et demande un certificat de la paire (URL, K_{pub}) à une autorité de certification,
   C'est tout côté développeur, le reste est transparent.
3. (K_{pub}, K_{priv}) est sauvegardée comme clefs-maîtresses du serveur,
4. à chaque demande de connexion d'un client, TLS génère une paire de clefs éphémères (K^{eph}_{pub}, K^{eph}_{priv}) adaptées pour des opérations sur les courbes elliptiques (pour ECDH),
5. une nouvelle clef de session symétrique K_{ab} est générée/partagée (*cf.* Slide 26),
6. au bout de quelques minutes, fin de session : nouvelles clefs éphémères ($K^{eph'}_{pub}$, $K^{eph'}_{priv}$), et nouvelle K'_{ab}.

Avantages de ECDHE

Deux avantages par rapport à un ECDH simple :

- forward-secrecy : connaître K_{ab} ne permet pas de trouver K'_{ab},
- le dev Web n'a pas besoin de générer une paire (K_{pub}, K_{priv}) adaptée pour faire des courbes elliptiques : c'est l'étape 4, gérée par TLS, qui génère des clefs éphémères pour les courbes elliptiques à partir des clefs du dev. En particulier, la plupart des dev ont une paire de clefs (K_{pub}, K_{priv}) pour faire du RSA, et le passage à ECDHE a tout de même été transparent pour eux.

Contenu Partie II : Échange de clefs

Kerberos ?

Objectif

- protocole d'authentification réseau basé sur TCP/IP avec tiers de confiance
- dans le projet Athena (MIT, IBM, DEC) pour des réseaux ouverts
- très utilisé
- que de la cryptographie symétrique
- **protocole d'échange de clefs secrètes !**

Acteurs

- client : possède sa clef secrète K_C
- serveur : possède sa clef secrète K_S
- Ticket-Granting Service (TGS) : clefs secrètes K_{TGS} et K_S
- Key-Distribution Center (KDC) : clefs secrètes K_{TGS} et K_C

Fonctionnement (métaphore)

Similaire à l'authentification par ticket au cinéma :

1. le client prend un ticket qui l'authentifie
2. l'ouvreur déchire le ticket en deux et chacun garde une partie
3. s'il faut contrôler, on vérifie que les déchirures correspondent
4. durée de vie limitée (dans Kerberos $\sim$ 8h)

Défauts

- failles mineures
- la sécurité repose sur le temps (dépendant de l'infrastructure et des OS)
- code public : permet la vérification

Usages

Vous vous servirez peut-être de Kerberos en pratique :

- utilisé pour des réseaux internes d'entreprise :
  - serveur central admin qui contient Kerberos,
  - toute machine du réseau interne gère ses clefs en se connectant au serveur Kerberos.
- souvent couplé à un **Active Directory**

Active Directory

Je n'ai pas le temps de faire un cours d'Active Directory (AD).

Active Directory

Annuaire central des comptes utilisateurs d'un réseau interne Windows.

- les utilisateurs ont un compte sur l'AD qui leur permet de se connecter sur n'importe quelle machine du S.I.
- l'AD gère les droits d'accès,
- l'AD gère beaucoup de logs (pour le monitoring),
- l'AD intègre souvent Kerberos pour la gestion des clefs d'utilisateurs.

Contenu Partie II : Échange de clefs

PKI : Infrastructure de Clefs Publiques

PKI

On appelle **PKI** une infrastructure qui permet de partager des associations (identité, clef publique) en garantissant (a) l'authentification de l'identité et (b) l'intégrité de la clef publique.

- cette infrastructure ne peut pas être cryptographique (c'est pour mettre en place le canal cryptographique, il n'existe pas encore),
- il existe de nombreuses alternatives, toutes reposent sur la **confiance**,
- Exemples :
  - certificat de clef publique (repose sur la confiance en une autorité de certification centrale), *e.g.* TLS,
  - PGP - Web of Trust (repose sur : l'ami de mon ami est mon ami, j'ai confiance),
  - ...

On étudiera en détails la PKI la plus importante : celle des **certificats électronique**.

Certificats électroniques

cf. Ch 4 : Introduction à la cryptographie

- partage de paire **(identité, clef publique)**
- **authentifie** *identité*, garantit l'**intégrité** de *clef publique*
- nécessite un **tiers de confiance** (autorité de certification CA)
- fournit un **document d'authentification** à durée limitée (certificat $\sim$ carte d'identité)
- pas besoin de consulter une base de donnée en temps réel pour valider le certificat

Utilisations :

- mél sécurisé (S-MIME)
- SSL/TLS : HTTPS, IMPAS, SMTP
- réseaux virtuels privés (IPsec)
- en remplacement d'un mot de passe

Standard de certificats électroniques

Norme de certificat X.509 (https://www.rfc-editor.org/rfc/rfc5280) :

- version
- numéro de série
- algorithme de signature du certificat
- nom du signataire du certificat
- date limite de validité
- détenteur du certificat
- **infos sur la clef publique** (algo, clef)
- identifiant du signataire
- **identifiant du détenteur**
- extensions
- signature des informations ci-dessus

Problématiques

Lors d'une demande de certificat :

- qui recueille/vérifie les informations ?
- quelles procédures ?
- qui crée le certificat ?
- qui le délivre ?
- quelle durée de validité ?
- où le stocker ?
- où récupérer le certificat d'un autre ?
- comment supprimer un certificat ?

⇒ **Public Key Infrastructure (PKI)**

Définition (PKI pour certificats électroniques)

Manipule des paires (K_{pub}, K_{priv}) et des certificats.

Composants :

- autorité de certification (CA)
- autorité d'enregistrement
- système de publication/distribution de certificats (annuaire)
- applications compatibles avec la PKI

Autorité d'enregistrement

Autorité d'enregistrement

- vérifie l'identité du demandeur de certificat
- s'assure qu'il possède un couple (K_{pub}, K_{priv})
- récupère la clef publique
- transmet les informations à l'autorité de certification

Remarque : les autorités d'enregistrement et de certification se connaissent et communiquent de manière sécurisée

Autorité de certification

Autorité de certification

- délivre des certificats électroniques
- a son propre certificat (auto-signé ou signé par une autre CA)
- signe les certificats avec sa clef privée
- garante de l'identité du propriétaire du certificat délivré
- doit être digne de confiance
- envoie les certificats au service de publication

Exemples de CA

- BNP Paribas (Net Identity)
- CertEurope (CertEurope Classe 3+)
- CertiNomis (SociePoste)
- Crédit Agricole (CA Certificat)
- LCL (CL Authentis)

Service de publication

Service de publication

- rend disponibles les certificats émis par la CA
- publie la liste des certificats valides
- publie la liste des certificats révoqués
- annuaire LDAP ou serveur Web

D'après vous ?

Comment faire pour vérifier la validité du certificat ?

D'après vous ?

Comment faire pour vérifier la validité du certificat ?

$\Rightarrow$ il faut vérifier la validité de la signature : *la CA a signé le certificat* avec sa **clef privée K^{CA}_{priv}, on peut vérifier que le certificat est valide en vérifiant la signature avec la clef K^{CA}_{pub}.**

D'après vous ?

Comment obtenir la clef publique K_{pub}^{CA} ?

D'après vous ?

Comment obtenir la clef publique K_{pub}^{CA} ?

Deux cas :

- C'est une CA interne à l'entreprise, pour des communications sur le réseau interne. Alors la clef publique de la CA doit être installée dans toutes les machines du réseau interne **lors de leur déploiement**.
- C'est une CA pour communiquer sur Internet. Alors la clef publique de toute CA majeure est installée lors **de l'installation du système d'exploitation**.

En pratique pour les certificats TLS (résumé)

Mise en place de certificat pour TLS pour Internet

- la CA crée sa paire (K^{CA}_{pub}, K^{CA}_{priv}),

En pratique pour les certificats TLS (résumé)

Mise en place de certificat pour TLS pour Internet

- la CA crée sa paire $(K^{CA}_{pub}, K^{CA}_{priv})$,
- tout système d'exploitation intègre K^{CA}_{pub}, tout navigateur Internet reconnaît K^{CA}_{pub},

En pratique pour les certificats TLS (résumé)

Mise en place de certificat pour TLS pour Internet

1. la CA crée sa paire $(K^{CA}_{pub}, K^{CA}_{priv})$,
2. tout système d'exploitation intègre K^{CA}_{pub}, tout navigateur Internet reconnaît K^{CA}_{pub},
3. un dev Web crée sa paire $(K^{url}_{pub}, K^{url}_{priv})$

En pratique pour les certificats TLS (résumé)

Mise en place de certificat pour TLS pour Internet

1. la CA crée sa paire $(K^{CA}_{pub}, K^{CA}_{priv})$,
2. tout système d'exploitation intègre K^{CA}_{pub}, tout navigateur Internet reconnaît K^{CA}_{pub},
3. un dev Web crée sa paire $(K^{url}_{pub}, K^{url}_{priv})$
4. il obtient un certificat de l'association (url, K^{url}_{pub}) de la part de la CA.
   Ce certificat est signé avec K^{CA}_{priv}.

En pratique pour les certificats TLS (résumé)

Mise en place de certificat pour TLS pour Internet

1. la CA crée sa paire $(K^{CA}_{pub}, K^{CA}_{priv})$,
2. tout système d'exploitation intègre K^{CA}_{pub}, tout navigateur Internet reconnaît K^{CA}_{pub},
3. un dev Web crée sa paire $(K^{url}_{pub}, K^{url}_{priv})$
4. il obtient un certificat de l'association (url, K^{url}_{pub}) de la part de la CA.
   Ce certificat est signé avec K^{CA}_{priv}.
5. quand un client se connecte au serveur, le serveur lui envoie (url, K^{url}_{pub}) signé avec K^{CA}_{priv},

En pratique pour les certificats TLS (résumé)

Mise en place de certificat pour TLS pour Internet

1. la CA crée sa paire $(K^{CA}_{pub}, K^{CA}_{priv})$,
2. tout système d'exploitation intègre K^{CA}_{pub}, tout navigateur Internet reconnaît K^{CA}_{pub},
3. un dev Web crée sa paire $(K^{url}_{pub}, K^{url}_{priv})$
4. il obtient un certificat de l'association (url, K^{url}_{pub}) de la part de la CA.
   Ce certificat est signé avec K^{CA}_{priv}.
5. quand un client se connecte au serveur, le serveur lui envoie (url, K^{url}_{pub}) signé avec K^{CA}_{priv},
6. le navigateur du client connaît K^{CA}_{pub}, qui permet de vérifier l'association (url, K^{url}_{pub})

En pratique pour les certificats TLS (résumé)

Mise en place de certificat pour TLS pour Internet

1. la CA crée sa paire (K^{CA}_{pub}, K^{CA}_{priv}),
2. tout système d'exploitation intègre K^{CA}_{pub}, tout navigateur Internet reconnaît K^{CA}_{pub},
3. un dev Web crée sa paire (K^{url}_{pub}, K^{url}_{priv})
4. il obtient un certificat de l'association (url, K^{url}_{pub}) de la part de la CA.
   Ce certificat est signé avec K^{CA}_{priv}.
5. quand un client se connecte au serveur, le serveur lui envoie (url, K^{url}_{pub}) signé avec K^{CA}_{priv},
6. le navigateur du client connaît K^{CA}_{pub}, qui permet de vérifier l'association (url, K^{url}_{pub})
7. le navigateur en déduit :

En pratique pour les certificats TLS (résumé)

Mise en place de certificat pour TLS pour Internet

1. la CA crée sa paire $(K^{CA}_{pub}, K^{CA}_{priv})$,
2. tout système d'exploitation intègre K^{CA}_{pub}, tout navigateur Internet reconnaît K^{CA}_{pub},
3. un dev Web crée sa paire $(K^{url}_{pub}, K^{url}_{priv})$
4. il obtient un certificat de l'association (url, K^{url}_{pub}) de la part de la CA.
   Ce certificat est signé avec K^{CA}_{priv}.
5. quand un client se connecte au serveur, le serveur lui envoie (url, K^{url}_{pub}) signé avec K^{CA}_{priv},
6. le navigateur du client connaît K^{CA}_{pub}, qui permet de vérifier l'association (url, K^{url}_{pub})
7. le navigateur en déduit :
   - qu'il se connecte bien à url,

En pratique pour les certificats TLS (résumé)

Mise en place de certificat pour TLS pour Internet

1. la CA crée sa paire $(K^{CA}_{pub}, K^{CA}_{priv})$,
2. tout système d'exploitation intègre K^{CA}_{pub}, tout navigateur Internet reconnaît K^{CA}_{pub},
3. un dev Web crée sa paire $(K^{url}_{pub}, K^{url}_{priv})$
4. il obtient un certificat de l'association (url, K^{url}_{pub}) de la part de la CA.
   Ce certificat est signé avec K^{CA}_{priv}.
5. quand un client se connecte au serveur, le serveur lui envoie (url, K^{url}_{pub}) signé avec K^{CA}_{priv},
6. le navigateur du client connaît K^{CA}_{pub}, qui permet de vérifier l'association (url, K^{url}_{pub})
7. le navigateur en déduit :
   - qu'il se connecte bien à url,
   - que K^{url}_{pub} est bien la clef publique de url.

En pratique pour les certificats TLS (résumé)

Mise en place de certificat pour TLS pour Internet

1. la CA crée sa paire $(K^{CA}_{pub}, K^{CA}_{priv})$,
2. tout système d'exploitation intègre K^{CA}_{pub}, tout navigateur Internet reconnaît K^{CA}_{pub},
3. un dev Web crée sa paire $(K^{url}_{pub}, K^{url}_{priv})$
4. il obtient un certificat de l'association (url, K^{url}_{pub}) de la part de la CA.
   Ce certificat est signé avec K^{CA}_{priv}.
5. quand un client se connecte au serveur, le serveur lui envoie (url, K^{url}_{pub}) signé avec K^{CA}_{priv},
6. le navigateur du client connaît K^{CA}_{pub}, qui permet de vérifier l'association (url, K^{url}_{pub})
7. le navigateur en déduit :
   - qu'il se connecte bien à url,
   - que K^{url}_{pub} est bien la clef publique de url.
8. avec cette clef publique, navigateur et serveur créent un canal sécurisé TLS.

Contenu Partie II : Échange de clefs

Structures de PKI

- structure hiérarchique
- structure croisée
- structure à pont
- structure décentralisée

Structure hiérarchique

Structure arborescente :

- racine de confiance : CA
- chaque CA donne des certificats à ses CA filles et à des utilisateurs
- confiance hiérarchique (en les CA au-dessus)
- évite d'avoir une seule CA en charge de tous les certificats
- chaînes de certificats (certificats entité $\rightarrow$ racine)

Structure croisée

- certificats obtenus par des CA racines distinctes
- chaque CA certifie la clef publique d'une autre CA
- un utilisateur certifié par CA1 peut vérifier le certificat d'un utilisateur de CA2
- s'il y a beaucoup de CA, ça devient une infrastructure lourde

Structure à pont

- évite une infrastructure lourde
- ajout d'une CA de pont, liée à toutes les CA racines
- le pont délivre des certificats entre CA racines, mais pas vers des utilisateurs

Structure décentralisée (Web of Trust)

Idée de PGP (Pretty Good Privacy) :

- protection de mél gratuit par Ph. Zimmermann
- utilise :
  - IDEA pour chiffrement
  - RSA pour la gestion de clefs et les signatures
  - MD5 pour le hachage cryptographique
- pas de CA

$\Rightarrow$ "climat de confiance"

- chaque utilisateur crée et distribue sa propre clef publique
- clefs publiques signées mutuellement
- chacun choisit à qui faire confiance

Exemple :

A et B sont amis, B et C sont amis.

Si A veut parler à C, il utilise le certificat émis par B.

$\sim$ plus on est proche de quelqu'un, plus on a confiance