Table of contents

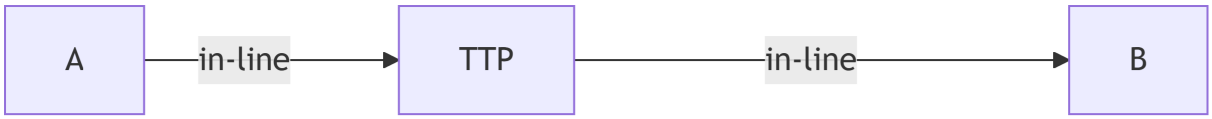
Tiers de Confiance et Certification (Trusted Third Parties - TTP)

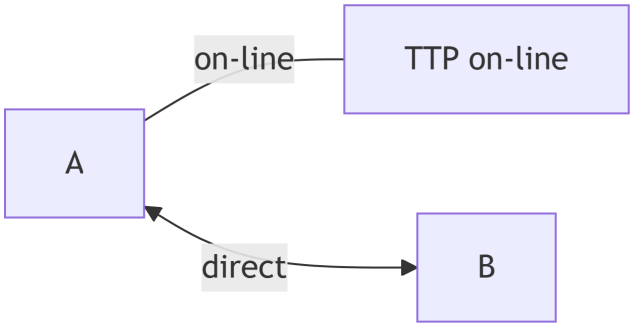
TTP

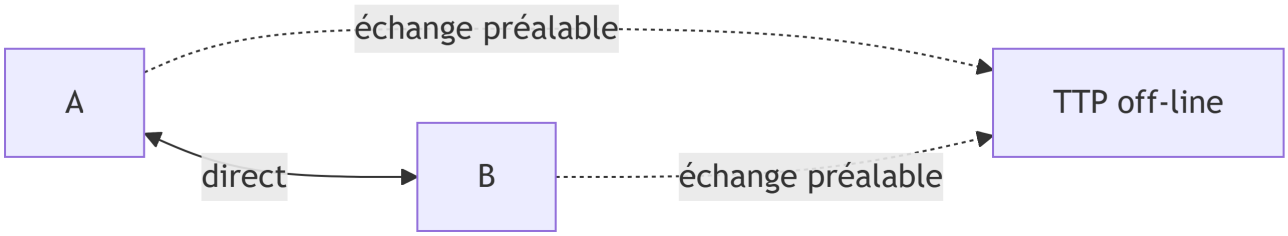
Modes de Fonctionnement des TTP

Trois modes opérationnels :

- **In-line** : TTP agit comme intermédiaire, relaye tous les échanges en temps réel (ex: Proxies, Secure Gateways)
- **On-line** : TTP participe en temps réel mais A et B communiquent directement (ex: KDC)
- **Off-line** : TTP ne participe pas en temps réel, rend l'information disponible à priori (ex: CA)







Comparaison :

- **Off-line** : Échanges facilités, pas besoin de disponibilité permanente
- **In-line/On-line** : Disponibilité permanente requise
- **Off-line** : Révocation des privilèges plus complexe

Texte original

TTP: Modes de Fonctionnement

- **In-line** : Le TTP agit comme **intermédiaire** pour relayer en temps réel les échanges entre A et B. Exemples : Proxies, Secure Gateways.
- **On-line** : Le TTP **participe en temps réel** aux échanges entre A et B mais A et B communiquent directement (sans passer par le TTP). Exemple : Key Distribution Center.
- **Off-line** : Le TTP **ne participe pas** à l'échange en temps réel mais rend l'information disponible **à priori**. Exemple : Certification Authorities.

Comparaison In-line/On-Line/Off-line : Échanges facilités et pas besoin de disponibilité permanente des TTP dans le off-line (contrairement aux deux autres) mais **révocation des privilèges** (p.ex. : lorsqu'une clé secrète est compromise) plus complexe.
[Diagrammes montrant les trois modes]

Révision rapide

Modes TTP :

- **In-line** : intermédiaire relay
- **On-line** : participation temps réel
- **Off-line** : info à priori (ex: CA)
- Off-line : pas de disponibilité requise mais révocation complexe

---

Key Distribution Centers (KDCs)

Objectif : Résoudre le problème de distribution n^2 clés.

Principe :

- Sans KDC : $\frac{n(n-1)}{2} \approx n^2$ clés pour n entités
- Avec KDC : Seulement n clés (chaque entité partage une clé avec KDC)
- Clés de session générées dynamiquement par KDC

Avantages :

- Scalabilité : une nouvelle entité = une seule nouvelle clé
- Établissement canaux sûrs via tickets (à la Kerberos)

Inconvénients :

- **Single point of security failure** : KDC compromis → tout le système vulnérable
- **Single point of operational failure** : KDC indisponible (DoS) → système paralysé
- **Performance bottleneck** : Opérations coûteuses (encryption, random generation)

Solutions : Mirroring, répartition de charge

Texte original

TTP: Key Distribution Centers (KDCs)
But : Résoudre le **n² key distribution problem** :

- Dans un environnement symétrique de n entités sans intermédiaire : $n(n-1)/2 \sim n^2$ clés différentes sont nécessaires pour toutes les paires d'entités partagent une clé différente.
- De plus, un tel système n'est pas **évolutif** (scalable) car l'adjonction d'une entité se traduit par la génération de n nouvelles clés.

Si chaque entité partage une clé avec un KDC, seules **n clés** sont nécessaires pour le fonctionnement du système et une clé suffit pour chaque nouvelle entité. L'établissement de canaux sûrs étant assuré par la génération de clés de session et la présence des tickets à la Kerberos.
Problèmes :

- **Single point of security failure** : par construction le KDC peut usurper l'identité de tous les nœuds du réseau. S'il est compromis tout le système devient vulnérable.
- **Single point of operational failure** : le mode de fonctionnement habituel d'un KDC est on-line (év. in-line). S'il devient indisponible (p.ex. suite à un denial of service attack), tout le système est paralysé.
- **Performance bottleneck** : les opérations des KDC sont souvent coûteuses en temps de calcul (cryptage/décryptage, random generation, etc.). Des solutions classiques (ie. le mirroring) doivent être envisagées pour répartir la charge des KDCs.

Révision rapide

KDC :

- Résout problème $n^2 \rightarrow n$ clés
- Scalable : $+1$ entité $= +1$ clé
- Risques : single point of failure (sécurité + opérationnel), bottleneck

---

CA

Certification Authorities (CAs)

Rôle : Authentifier l'association entre une **entité** et sa **clé publique**.

Fonctionnement :

1. CA vérifie identité (passeport, etc.)
2. CA crée et signe **certificat** contenant cette association
3. Certificats accessibles aux entités (peuvent être cachés)

Vérification : Nécessite copie authentique de la clé publique de la CA.

Avantages :

- Mode **off-line** : Indisponibilité courte acceptable
- Simpler then safer : pas de protocoles complexes

Inconvénients :

- **Révocation asynchrone** : Certificat peut devenir invalide (vol clé privée)
- **Solution** : Certificate Revocation Lists (CRLs) signées

Compromission CA :

Conséquences graves si clé privée de signature compromise.

> Texte original
>
> **TTP: Certification Authorities (CAs)**
> Le rôle premier d'une **Entité de Certification** (Certification Authority ou **CA**) est d'**authentifier l'association entre une entité et sa clé publique** (pensez aux attaques Man-In-the-Middle !).
> La CA va **créer et signer des certificats** contenant cette association (moyennant une preuve d'identité comme un passeport) et les rendre accessibles aux entités concernées.
> Une fois signées, des copies des certificats (**cached certificates**) peuvent être gardées dans des endroits non protégés (p.ex. dans l'espace disque de l'utilisateur). Cependant, afin de vérifier la signature des certificats, les entités concernées nécessitent une **copie authentique de la clé publique de la CA**.
> **Simpler then safer** : pas besoin d'implanter des protocoles complexes dans une CA.
> Le mode de fonctionnement habituel d'une CA est **off-line**, ce qui diminue les conséquences des périodes (courtes…) d'indisponibilité.
> **Problème associé au mode off-line :** la validité des cached certificates peut être remise en question de manière "**asynchrone**" par un vol de clé privé.
> **Remède :** les CAs publient également des listes signées des certificats non valides (**Certificate Revocation Lists** ou **CRLs**).
> Le compromis d'une CAs a des conséquences moins évidentes mais presque aussi néfastes que celui d'un KDC surtout si la clé privée servant à signer des certificats est aussi compromise.

Révision rapide

CA :

- Authentifie association entité clé publique
- Signe certificats (mode off-line)
- CRLs pour révocations
- Compromission = conséquences graves

Proof of Possession (PoP)

Problème : Vérifier identité ne suffit pas, il faut aussi vérifier possession de la clé privée.

Attaque sans PoP :

1. A signe document et l'envoie à B (notaire) : $S_{priv_A}(\text{Invention}), S_{priv_{CA_A}}(Cert_A)$
2. C intercepte, demande à CA_C (sans PoP) certificat associant C à pub_A
3. C envoie à B : $S_{priv_A}(\text{Invention}), S_{priv_{CA_C}}(Cert_C)$
4. C devient l'inventeur !

Protocole PoP simple :

1. $CA \rightarrow A : A, r$ (nombre aléatoire + identité)
2. $CA \leftarrow A : S_{priv_A}(A, r)$
3. CA vérifie signature avec pub_A

Conséquences :

- Introduit **niveaux de confiance** pour CAs
- Critères : PoP, mise à jour CRLs, sécurité clé signature
- Problème aggravé par prolifération non contrôlée des CAs

Texte original

CA: Proof of Possession (PoP)
La vérification de l'identité de A pour créer (év. modifier) un certificat associant A à sa clé publique n'est pas un critère suffisant. Il faut également **vérifier que A possède vraiment la clé privée correspondante**.
Soient A et sa CA : CA_A. Voyons ce qu'un attaquant actif C peut faire en "collaboration" avec une CA_C qui ne vérifierait pas la PoP :
A signe un document contenant la description d'une invention révolutionnaire et l'envoie

à B (le notaire) avec son certificat signé par CA_A :
$A \rightarrow B : S_{priv_A}(\text{Invention}), S_{priv_{CA_A}}(Cert_A)$
C intercepte ce paquet, s'adresse à CA_C et lui demande de créer un certificat associant son identité C à la clé publique de A et envoie à B :
$C \rightarrow B : S_{priv_A}(\text{Invention}), S_{priv_{CA_C}}(Cert_C)$
C devient ainsi l'inventeur révolutionnaire...
Protocole simple de vérification de PoP :
$CA \rightarrow A : A, r$; r : nb. aléatoire, A pour protéger A des chosen message attacks.
$CA \leftarrow A : S_{priv_A}(A, r)$; CA n'a plus qu'à vérifier la signature avec pub_A.
Ce critère et d'autres critères de comportement comme la mise à jour des CRLs ou la sécurité de la clé de signature introduisent des **niveaux de confiance** pour les CAs et pour les certificats qu'elles signent.
Ce phénomène s'aggrave avec la **prolifération non contrôlée des CAs** !

Révision rapide

PoP (Proof of Possession) :

- Vérifier possession clé privée (pas juste identité)
- Sans PoP : attaquant peut usurper identité
- Protocole : CA vérifie signature sur challenge
- Introduit niveaux de confiance pour CAs

Séparation Certification/Révocation

Problème : Si même clé signe certificats et CRLs, adversaire possédant clé privée CA peut :

1. Publier CRL révoquant certificat de victime A
2. Créer faux certificat associant A à une clé contrôlée par adversaire
3. Décrypter transactions confidentielles pour A
4. Se faire passer par A (signatures, authentification)

Solution : Separation of Duties

Séparation claire des tâches :

- **Certificats** et **CRLs** signés avec **clés différentes**
- Par **entités fonctionnelles différentes** :
  - **Certification Authority** (CA)
  - **Revocation Authority** (RA)

- Résidant dans **machines différentes**
- Soumises à **security policies indépendantes**

Texte original

CA: Certification et Révocation
Problème : Si la même clé sert à signer les certificats et les CRLs, un adversaire possédant la clé privée de signature d'une CA peut attaquer une "victime" A sous l'autorité de cette CA comme suit :

- Publier une CRL contenant le certificat révoqué de A.
- Créer un certificat associant A à une clé publique dont il contrôle la clé privée pour ensuite :
  - jouer le Man-In-the-Middle pour décrypter les transactions confidentielles pour A ;
  - se faire passer par A pour des transactions authentifiées ou des documents signés.

Solution : Separation of duties : La certification et la révocation deviennent des tâches clairement différenciées :

- Certificats et CRLs sont signés avec des **clés différentes**,
- par des **entités fonctionnelles différentes** (Certification Authority et Revocation Authority) ;
- si possible, résidant dans des **machines différentes** soumises à des critères de sécurité (security policies) **indépendants**.

Révision rapide

Separation of Duties :

- Certificats CRLs (clés différentes)
- CA Revocation Authority
- Machines et policies séparées
- Évite attaques post-compromission

Entités Fonctionnelles Liées à la Certification

Name Server :

- Gestion espace de noms unique et cohérent
- Combiné avec certification si authentification nécessaire
- Exemple : **DNSSec** (DNS authentifié pour Internet)

Registration Authority (RA) :

- Tâches nécessitant contact direct avec entités
- Vérification identité, PoP, demandes/modifications certificats
- Détachée de CA pour raisons géographiques

Key Generator :

- Génération de paires clés publique/privée
- **Avantages** : Simplicité utilisateurs, sécurité renforcée
- **Désavantage** : Clé privée connue d'une autre entité → perte non-répudiation

Certificate Directory :

- Répertoire accès lecture seule aux certificats

Texte original

Entités Fonctionnelles Liées à la Certification

- **Name Server** : responsable de la gestion d'un **espace de noms unique et cohérent**. Lorsque l'authentification est nécessaire, la gestion des noms doit être complétée par la certification des clés publiques associées à ces noms.

  Exemple d'une solution pilote combinant les deux concepts : **DNSsec** : environnement de gestion de noms authentifiés pour Internet.

- **Registration Authority** : Entité chargée d'accomplir les tâches relatives à la gestion des certificats nécessitant un **contact direct** avec les entités concernées. Ces tâches comprennent la vérification des paramètres nécessaires à la demande initiale ou à la modification des certificats (vérification d'identité, PoP, etc.). Le fait de détacher cette fonctionnalité de la CA est normalement dû à des considérations géographiques.

- **Key Generator** : Permet de déléguer le processus de création de paires de clés publique/privée à une entité dédiée :
  - **Avantages** : simplicité pour les utilisateurs ; possibilité de renforcer la sécurité des paires choisies.
  - **Désavantage** : Clé privée connue d'une autre entité ! Perte de la non-répudiation.

- **Certificate Directory** : Le répertoire permettant aux utilisateurs d'accéder (en lecture seulement) aux certificats des correspondants.

Révision rapide

Entités certification :

- **Name Server** : noms + DNSSec
- **RA** : contact direct, vérifications
- **Key Generator** : génération clés (perd non-répudiation)
- **Certificate Directory** : accès lecture certificats

Autres TTPs

Timestamp Agent (TA) :

- Certifie existence document/transaction à un moment précis
- Méthodes :
  - Associer timestamp au document (ou h(doc)) + signer
  - Utiliser authentication tree (arbre d'authentification)

Notary Agent :

- Comme TA mais aussi : validité, origine, appartenance
- Support légal pour non-répudiation

Key Escrow Agent (KEA) :

- Accès clés secrètes de session sous conditions (mandat judiciaire)
- Nécessite système cryptage dédié

Exemple : Clipper/Capstone

- **Clipper chip** (1993) : Encryption symétrique avec accès KEA
- Polémique, failles découvertes
- **Capstone chip** : Successeur (carte PCMCIA Fortezza)
- Military level security

Texte original

Autres TTPs

- **Timestamp agent (TA)** : Certifie l'existence d'un document ou le déroulement d'une transaction à un moment bien spécifié dans le temps. Pour ce faire le TA peut :
  - associer un timestamp au document (ou à h(doc) avec h une Collision Resistant Hash Fonction) et signer le tout avec sa clé privée et
  - utiliser un authentication tree (arbre d'authentification, cf. page 231).
- **Notary agent** : Certifie non seulement l'existence d'un document à un temps donné (comme le TA) mais également sa **validité, origine ou appartenance** à une entité donnée. Ce service constitue un support (légalement nécessaire ?) pour la non-répudiation.
- **Key escrow agent (KEA)** : Entité autorisée à accéder aux clés secrètes de session pourvu que certaines conditions (p.ex. un mandat judiciaire) soient remplies. Ceci nécessite un système de cryptage dédié. Exemple, le Clipper key escrow system :
  - Annoncé en Avril 1993 par l'administration USA, au milieu d'une grande polémique, comme la solution de cryptage de communications à grande échelle.
  - Le **Clipper chip** est un dispositif de cryptage/décryptage symétrique donnant accès aux clés des session lorsque les clés secrètes de deux KEAs (normalement des agences fédérales) lui sont fournies en entrée.
  - La présence de quelques failles ainsi que le besoin de crypto asymétrique ont donné lieu à son successeur : le **Capstone chip** pouvant être intégré dans une carte PCMCIA (appelée **Fortezza** et utilisée pour military level security).

Révision rapide

Autres TTPs :

- **TA** : timestamp existence document
- **Notary** : TA + validité/origine (non-répudiation)
- **KEA** : accès clés sous conditions légales
- Exemple : Clipper/Fortezza (controversé)

Public Key Authentication

Certificats

Définition : Information associant une entité à sa clé publique.

Structure générique :

- **Serial Number, Version**
- **Issuer** : Identité CA signataire (globale et unique)
- **Signature Algorithm** : Algorithme calcul signature (ex: MD5+ElGamal, SHA+RSA)
- **Subject** : Nom entité certifiée (global et unique)
- **Subject Public Key** : Clé publique
  - RSA : (n, e)
  - DH : $(p, \alpha, \alpha^x \mod p)$
- **Subject Public Key Algorithm** : RSA, DH, etc.
- **Validity** : Période validité (UTC)
- **Signature** : Porte sur tous enregistrements précédents, garantit authenticité

Texte original

Authenticité des clés publiques: Certificats
Un **certificat** est une pièce d'information associant une entité à sa clé publique. De manière générique, il est constitué des élément suivants :

- **Serial Number, Version**.
- **Issuer** : l'identité (global et unique) de la CA signataire.
- **Signature Algorithm** : l'algorithme permettant de calculer la signature sur le certificat. P.ex. : MD5 + ElGamal ou SHA + RSA.
- **Subject** : Le nom (global et unique) de l'entité dont la clé publique est certifiée.
- **Subject Public Key** : La clé publique de l'entité. Par exemple :
  - (n, e) : modulus et exposant publique pour RSA.
  - $(p, \alpha^x \mod p)$: modulus, générateur et partie publique pour Diffie-Hellman.
- **Subject Public Key Algorithm** : L'algorithme associé à la clé publique. P.ex : RSA ou Diffie-Hellman.
- **Validity** : La période de validité du certificat, normalement exprimée en UTC.
- **Signature** : Contient la signature effectuée au moyen du Signature Algorithm et de la clé privée de la CA. Elle porte sur l'ensemble des enregistrements précédents et garantit ainsi l'authenticité des informations qu'ils contiennent.

Révision rapide

Certificat :

- Issuer (CA) + Subject (entité)
- Clé publique + algorithme
- Validity period
- Signature CA sur tout

Certificate Revocation Lists (CRLs)

Définition : Listes de certificats devenus invalides.

Raisons révocation :

- Clé privée compromise
- Modification algorithme
- Changement fonction (role-based certificate)
- Autres facteurs invalidant informations certificat

Structure CRL :

- **Issuer, Signature Algorithm**
- **Date of Issue, Date of Next Issue**
- Pour chaque certificat révoqué :
  - **Serial Number**
  - **Revocation Date**
- **Signature** : Sur toute la liste

Exigences :

- CAs doivent publier CRLs **fréquemment**
- Canaux distribution **large audience**
- Minimiser risque fraudes

Problème : Révocation = talon d'Achille systèmes à clés publiques

Solutions alternatives :

- Certificats validité très courte (quelques minutes)
- Re-confirmation périodique par CAs

- Retour au mode on-line → haute disponibilité requise

Texte original

Certificate Revocation Lists (CRLs)
Il s'agit de listes contenant des certificats devenus **non valables** suite à une clé privée compromise ou à tout autre facteur mettant en évidence la validité des informations contenues dans un certificat (changement de l'algorithme utilisé, changement de fonction pour un role-based certificate, etc.).
Une CRL générique a les éléments suivants :

- **Issuer, Signature Algorithm** : comme pour les certificats.
- **Date of Issue, Date of Next Issue** : date d'émission et date de la prochaine émission.
- Pour chaque certificat révoqué, les enregistrements suivants :
  - **Serial Number** du certificat révoqué.
  - **Revocation Date**.
- **Signature** : signature portant sur toute la liste.

Une CA se doit de publier des CRLs avec une **fréquence très élevée** et en utilisant des canaux de distribution de **large audience**, afin de diminuer le risque de fraudes.
La révocation est le talon d'Achille de tout système à clés publiques...
Une solution : certificats avec des lapses de validité très courts (quelques minutes) exigeant une re-confirmation périodique de la part des CAs...
...mais ceci nous fait revenir au mode on-line et à imposer, donc, une grande disponibilité de la part des CAs.

Révision rapide

CRLs :

- Listes certificats invalides (compromis, etc.)
- Structure : issuer, dates, serial numbers, signature
- Publication fréquente requise
- Talon d'Achille PKI
- Alternative : certificats courte durée (→ on-line)

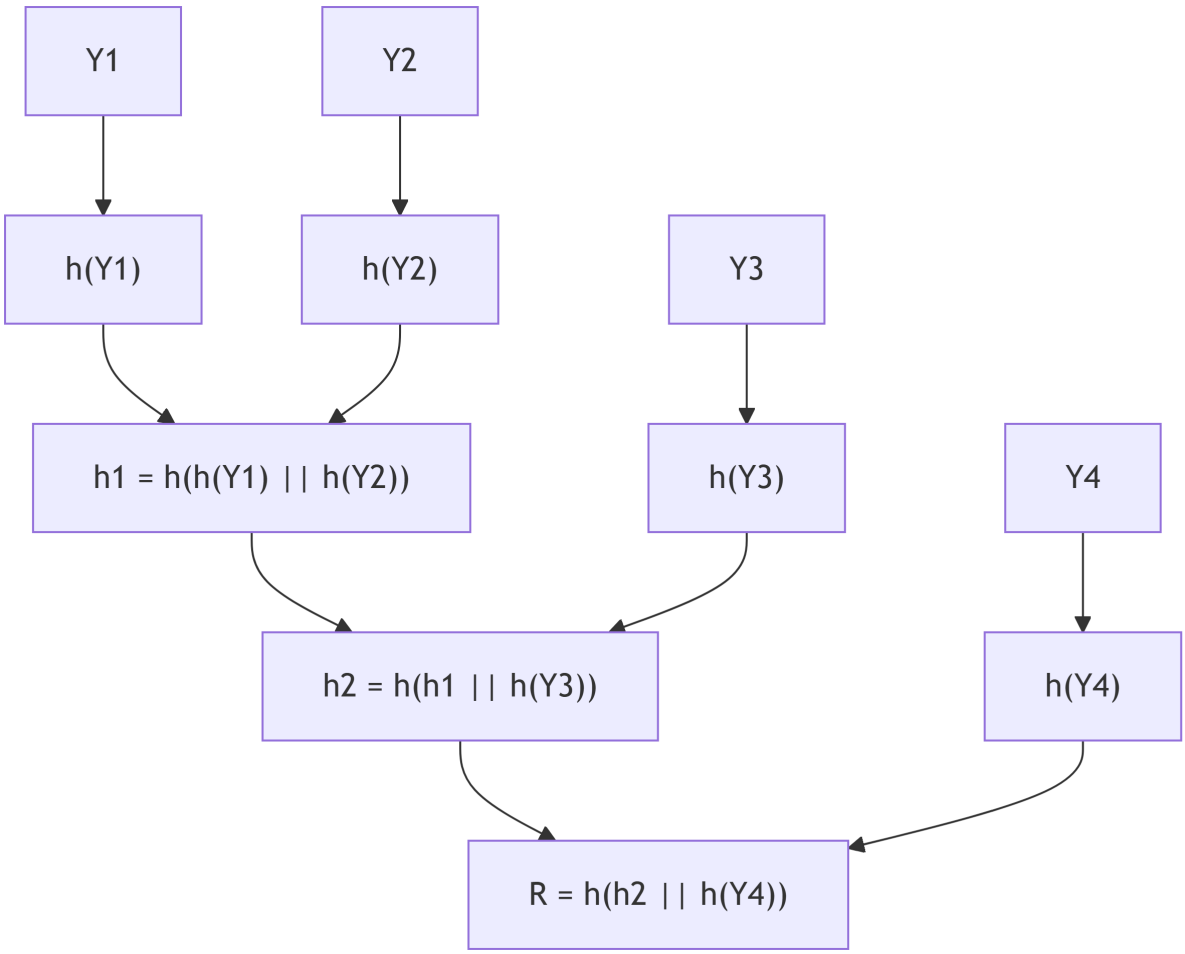
Arbres d'Authentification

Principe : Alternative à certification pour authentifier informations publiques.

Construction : Arbre (binaire) avec hash fonction + authentification racine.

Pour n valeurs $Y_1, Y_2, ..., Y_n$:

1. Valeurs placées dans feuilles
2. Arcs depuis feuilles étiquetés $h(Y_i)$
3. Nœuds non-terminaux étiquetés $h(h_1||h_2)$ ($||$ = concaténation)
4. Racine R authentifiée (signature digitale)



Vérification Y_1 **:** Fournir $h(Y_2), h(Y_3), h(Y_4)$, calculer $h(Y_1), h_1, h_2$ et vérifier $h(h_2||h(Y_4)) = R$

Avantages :

- Seul R nécessite protection cryptographique
- Arbres équilibrés : données intermédiaires $\sim \log_2 n$

Inconvénients :

- Nœud modifié $\rightarrow$ recalcul chemin jusqu'à racine
- Ajout nœuds $\rightarrow$ arbres non-équilibrés recommandés

Application principale : Timestamping

TA

- construit arbre
- fournit timestamp signé + chemin vérification
- publie R quotidiennement (journal).

Texte original

Arbres d'Authentification
Les arbres d'authentification sont une **alternative à la certification** pour authentifier des information publiques.
Il s'agit d'exploiter les avantages d'une structure d'arbre (normalement binaire) avec l'utilisation de hash fonctions et l'authentification du nœud racine.
Soit un arbre A avec n feuilles. Soit h une collision resistant hash function (CRHF). L'arbre A peut être utilisé pour l'authentification de n valeurs publiques $Y_1, Y_2, ..., Y_n$ en construisant un arbre d'authentification comme suit :

1) Les valeurs $Y_1, Y_2, ..., Y_n$ sont placées dans les feuilles de l'arbre.
2) Chaque arc partant d'une feuille Y_i est étiqueté $h(Y_i)$ (h étant une CRHF).
3) Chaque nœud non-terminal ayant des arcs sous-jacents étiquetés h_1 et h_2 est étiqueté $h(h_1||h_2)$ (|| dénote concaténation).

[Diagramme arbre]
Pour vérifier l'authenticité de Y_1, il est nécessaire de fournir les valeurs $h(Y_2), h(Y_3), h(Y_4)$. Après, il suffit de calculer $h(Y_1), h_1$ et h_2 (selon la figure) et accepter l'authenticité de Y_1 si $h(h_2||h(Y_4)) = R$. Une modification illicite dans Y_1 se traduirait (par les caractéristiques de la CRHF) en une valeur différente pour $h(h_2||h(Y_4)) \neq R$.
À noter que seule la valeur R doit être authentifiée (p.ex. à l'aide d'une signature digitale). Les autres valeurs sont protégées par la non-réversibilité de la CRHF.
Avantage : Seul R nécessite une protection cryptographique pour l'authentification !
Inconvénients :

- Pour vérifier la valeur Y_1, les valeurs $h(Y_{2,3,4})$ et la valeur R sont nécessaires. Pour minimiser cet effet, on peut d'utiliser des arbres équilibrés (des arbres dont les

chemins différent d'au plus un arc) afin de réduire le nombre de données intermédiaires à $\sim \log_2 n$.
- Lorsqu'un nœud est modifié, tout le chemin jusqu'à la racine doit être re-calculé.
- Lorsque des nouveaux nœuds sont rajoutés, il convient de construire des arbres non-équilibrés (comme celui de la figure) et de rajouter les nœuds par la racine.

Application principale : timestamping : le timestamping agent (TA) construit un tel arbre et fournit au requérant le timestamp signé avec sa clé privée ainsi que le chemin de vérification. TA publie R quotidiennement dans un journal ce qui lui empêche de tricher !

Révision rapide

Arbres d'authentification :

- Alternative certification via hash + arbre
- Seule racine R signée
- Vérification : chemin $\sim \log_2 n$ valeurs
- Application : timestamping
- TA publie R quotidiennement

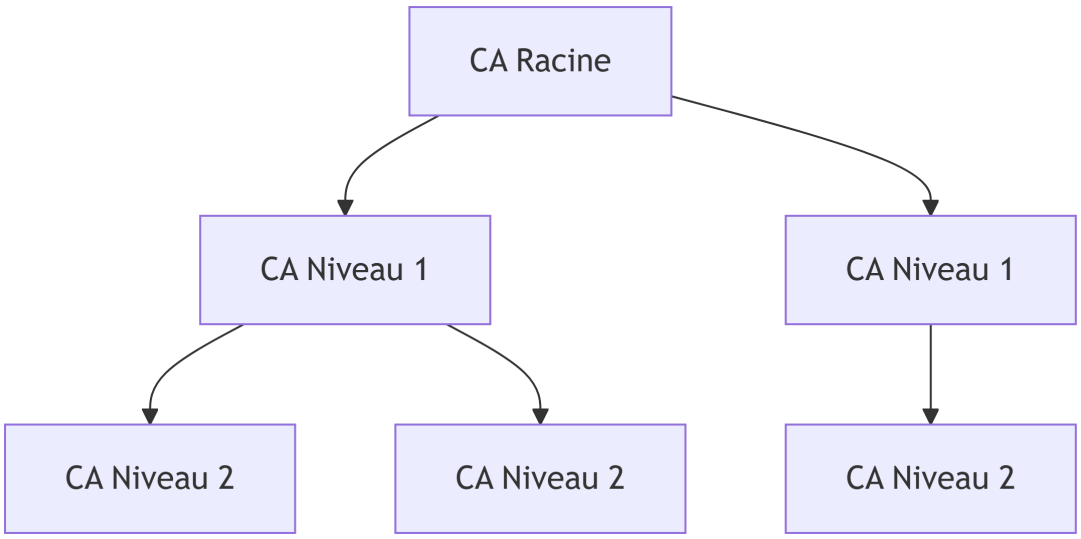
Topologies de Certification

Problème : Communication entre utilisateurs de CAs différentes → question de confiance.

Cross-Certification :

- CA_A certifie clé publique pub_{CA_B} de CA_B
- Certificat croisé : $CA_A\{CA_B\}$
- Chaîne de certification : $CA_A\{CA_B\} \rightarrow CA_B\{B\}$
- A vérifie clé B avec copie authentique pub_{CA_A}

Modèle Hiérarchique Strict (PEM/X.509) :



- Toute chaîne non-locale commence au nœud racine
- Clé publique racine supposée connue mondialement
- **Problème :** Centralisation, point de défaillance unique

Modèle Graphe (PGP) :

- Utilisateurs agissent comme CAs pour correspondants
- Structure graphe décentralisée
- Adapté groupes fermés
- **Limites :** Populations non connectées

Modèles Hybrides :

- Hiérarchie + certification croisée bidirectionnelle

Règle d'or : Chaînes courtes (maillon le plus faible !)

Texte original

Topologies de Certification
Lorsque deux utilisateurs appartenant à des CAs différentes souhaitent communiquer, il apparaît un problème de confiance : doit-on faire confiance à un certificat émis par une autre CA ?.
Le processus de **certification croisée** (cross-certification) permet à CA_A de certifier la clé publique pub_{CA_B} de CA_B. Le certificat résultant s'appelle certificat croisé (cross-certificate), on le note : $CA_A\{CA_B\}$.
Si A désire vérifier l'authenticité de la clé publique de B et il existe un certificat croisé

$CA_A\{CA_B\}$, A va demander à B de lui fournir son certificat signé par CA_B, soit $CA_B\{B\}$.
La **chaîne de certification** résultante : $CA_A\{CA_B\} \rightarrow CA_B\{B\}$ permet à A de vérifier la clé publique de B en utilisant une copie authentique de pub_{CA_A}.
La relation de confiance nécessaire à la certification croisée n'est pas toujours facile à établir dans des environnements concurrents, c'est pourquoi des **modèles hiérarchiques** entre les CAs ont été proposés. Exemple le modèle hiérarchique strict de PEM/X.509 :
[Diagramme hiérarchie]
Dans l'environnement PEM, toute chaîne de certification non-locale commence au **nœud racine**, dont la clé publique est supposée connue du monde entier…
D'autres modèles comme celui proposé par **PGP** se basent sur une structure de graphe où les nœuds sont les utilisateurs qui agissent comme CAs pour certifier les clés publiques des correspondants. Même si bien adapté pour des groupes fermés d'utilisateurs, ce modèle a ses limites lorsqu'il est appliqué à des populations non connectées.
D'autres schémas proposés combinent la structure hiérarchique avec la certification croisée bidirectionnelle.
Il faut garder les chaînes de certification **aussi courtes que possible** (une chaîne est toujours aussi vulnérable que son **maillon le plus faible** !).

Révision rapide

Topologies certification :

- **Cross-certification** : $CA_A\{CA_B\} \rightarrow CA_B\{B\}$
- **Hiérarchique** (PEM/X.509) : racine universelle
- **Graphe** (PGP) : utilisateurs = CAs
- **Hybride** : hiérarchie + cross-cert
- Règle : chaînes courtes !

Public Key Infrastructure (PKI)

Définition : Infrastructure intégrée fournissant services de sécurité basés sur cryptographie à clés publiques.

Entités Fonctionnelles Principales

Certification Authority (CA) :

- Création et maintenance certificats

Certificate Repository :

- Répertoire certificats accessible (X.500, LDAP, WWW, DNS)

Certificate Revocation :

- Gestion certificats compromis/obsolètes (CRLs)

Key Backup and Recovery :

- Sauvegarde/rétablissement clés perdues
- Destruction support, oubli password, départ employé
- Clé privée décryption (pas signature)

Automatic Key Update :

- Mise à jour clés après fin validité

Key and Certificate History :

- Récupération clés obsolètes pour décrypter anciens documents

Cross-Certification :

- Validation certificats d'autres PKIs (clients, fournisseurs, partenaires)

Support Non-Répudiation :

Démontre le bon déroulement d'une transaction authentifiée

- Data origin authentication
- Time-stamped data signature
- Signed receipt of delivery

Secure Time Stamping :

- Temps référence accepté par tous

Logiciel Client :

- Opérations PKI côté client
- Gestion certificats, signatures, décryption
- Périphériques (cartes à puces, biométrie)

Texte original

Public Key Infrastructure (PKI): Définitions
Définition : Une PKI est une infrastructure intégrée permettant de fournir un ensemble

de services de sécurité sur la base de la cryptographie à clés publiques.
Entités Fonctionnelles :

- **Entité de certification (Certification Authority ou CA)** : Entité responsable de la création et maintenance des certificats.
- **Répertoire des certificats (Certificate Repository)** mettant les certificats à disposition des utilisateurs et des applications. Technologies utilisées : X.500, LDAP, Serveurs WWW, DNS, etc.
- **Révocation des certificats (Certificate Revocation)** compromis ou devenus obsolètes (notamment gestion des CRLs)
- **Sauvegarde et rétablissement centralisés des clés (Key Backup and Recovery)** : Entité permettant de gérer la perte de clés suite à des événements divers : destruction du support matériel, oubli du mot de passe de déblocage, départ de l'employé, etc. À noter que cette procédure s'applique principalement à la clé privée de décryption (par opposition à la clé privée de signature).
- **Mise à jour automatique des clés (Automatic Key Update)** après la fin de leur validité.
- **Historique des clés et des certificats (Key and Certificate History)**. Cette entité permet de récupérer des clés devenues obsolètes, ayant servi à encrypter un document dans le passé.
- **Certification croisée (Cross-Certification)** avec d'autres PKI (clients, fournisseurs, partenaires, etc.). Cette fonctionnalité permet (sous certaines contraintes) de valider les certificats émis par d'autres PKIs
- **Support pour la non-répudiation** : Service à valeur ajouté permettant de fournir l'évidence nécessaire à démontrer le déroulement d'une transaction authentifiée (data origin authentication, time-stamped data signature, signed receipt of delivery, etc.)
- **Secure Time Stamping** : Entité capable de fournir un temps de référence accepté par tous les intervenants d'une PKI. Applications principales : non-répudiation, arbitrage en cas de conflits, etc.
- **Logiciel Client** : Cette entité fonctionnelle permet de réaliser toutes les opérations propres à la PKI côté client. Exemples : gestion des certificats utilisateurs, signature de documents, décryption d'information, gestion de périphériques spécifiques (lecteurs de cartes à puces, dispositifs biométriques, etc.)

Révision rapide

PKI - Entités principales :

- CA : création/maintenance certificats
- Repository : stockage certificats
- Revocation : CRLs
- Backup/Recovery : clés perdues (décryption)
- Cross-cert : validation autres PKIs
- Time stamping : référence temps
- Client : opérations utilisateur

Avantages et Inconvénients

Avantages :

Sécurité :

- Environnement intégré sans maillons faibles

Tout en un :

- Intégration multiples services : authentification forte, signatures, single sign-on, VPNs, B2C/B2B
- Économie vs solutions "au cas par cas"

Interopérabilité :

- Standards répandus (X.509, PKCS, OCSP)
- Applications et dispositifs compatibles
- Interopérabilité inter-entreprise possible

Inconvénients :

Coût de mise en place :

- Produits chers
- Compétences rares

Complexité :

- Mise en œuvre et gestion complexes
- **Alternative :** Sous-traitance service PKI

Texte original

PKI: Principaux Avantages et Inconvénients
Avantages

- **Sécurité** : La nature intégrée d'une PKI permet de créer un environnement de sécurité sans maillons faibles.
- **Tout en un** : Une PKI permet l'intégration et la gestion de tous les paramètres de sécurité propres à un grand nombre de services : authentification forte d'entités, signature des documents permettant la non-répudiation, single sign-on, réseaux privés virtuels (VPNs), communications sécurisées avec des clients/partenaires/fournisseurs (B2C, B2B), etc. La PKI constitue une économie notable par rapport aux solutions "au cas par cas".
- **Inter-opérabilité intra et inter entreprise** : Les principaux produits PKI répondent à des normes de standardisation très répandues (X.509, PKCS, OCSP, etc.). Un grand nombre d'applications et dispositifs matériels sont désormais conformes à ces standards. La compatibilité possible entre différents fournisseurs de PKIs permet également (sous quelques réserves) l'inter-opérabilité inter-entreprise.

Inconvénients

- **Coût de mise en place** : produits chers, compétences rares
- **Complexité**...mais :
  - la sous-traitance du "service" PKI est une alternative.

Révision rapide

PKI :
Avantages :

- Sécurité intégrée
- Tout-en-un : multi-services
- Interopérabilité (standards)

Inconvénients :

- Coût élevé
- Complexité
- Solution : sous-traitance