Chiffrement asymétrique

Chiffrement asymétrique

Chiffrement asymétrique

Le problème d'un chiffrement symétrique : **les entités légitimes doivent partager une clé secrète**.

Solution : le chiffrement asymétrique : la clé de chiffrement $\mathbf{k}_{pub}$ est différente de la clé de déchiffrement $\mathbf{k}_{priv}$.

- $\mathbf{k}_{priv}$ peut déchiffrer un message qui a été chiffré avec la clé $\mathbf{k}_{pub}$
- On ne peut pas déchiffrer avec seulement la connaissance de $\mathbf{k}_{pub}$. En particulier, on ne peut pas déduire $\mathbf{k}_{priv}$ de $\mathbf{k}_{pub}$.

Chiffrement asymétrique

Définition : Les fonctions à sens unique à trappe

Le système de chiffrement asymétrique est une fonction à sens unique à trappe.

- La clé de chiffrement $\mathbf{k}_{pub}$ et la clé de déchiffrement $\mathbf{k}_{priv}$ sont liées puisque l'une déchiffre le message qui a été chiffré avec l'autre.
- Le chiffrement avec $\mathbf{k}_{pub}$ se fait en temps polynomial.
- Le déchiffrement sans la clé $\mathbf{k}_{priv}$ se fait en temps exponentiel (idéalement, équivalent à un problème NP-complet).
- Le déchiffrement avec la clé $\mathbf{k}_{priv}$ se fait en temps polynomial : $\mathbf{k}_{priv}$ est appelée la trappe.

Chiffrement asymétrique

Exemple de fonction à trappe :

- Soit $f : x \longmapsto x^3 \pmod{100}$
- Connaissant x, trouver $f(x)$ est facile.
- Connaissant $y = f(x)$, trouver x est difficile ...
- ... sauf avec une trappe : $y \longmapsto y^7 \pmod{100}$

Chiffrement asymétrique

Alice, Albert, Alfred, ...

Bob

génère $\mathbf{k}_{pub}$ et $\mathbf{k}_{priv}$

$\mathbf{k}_{pub}$

ne connait que $\mathbf{k}_{pub}$

$\mathbf{m}$: message

$\mathbf{c} = Enc_{\mathbf{k}_{pub}}(\mathbf{m})$

$\tilde{\mathbf{c}}$

peut être différent de $\mathbf{c}$ *si Eve intercepte et modifie les messages.*

$\tilde{\mathbf{m}} = Dec_{\mathbf{k}_{priv}}(\tilde{\mathbf{c}})$

Rappels d'arithmétique

Rappels d'arithmétique

Division euclidienne

Théorème : Division euclidienne dans $\mathbb{N}$

Soient $a, b \in \mathbb{N}$ avec $b \neq 0$. Alors, il existe $q, r \in \mathbb{N}$, uniques, tels que :

$$a = bq + r \quad \text{avec } 0 \leq r < b$$

Corollaire : Division euclidienne dans $\mathbb{Z}$

Soient $a \in \mathbb{Z}$ et $b \in \mathbb{N}$ avec $b \neq 0$. Alors, il existe $q \in \mathbb{Z}$ et $r \in \mathbb{N}$, uniques, tels que :

$$a = bq + r \quad \text{avec } 0 \leq r < b$$

Définition

Soient $a, b \in \mathbb{Z}$. On dit que b **divise** a et on écrit $b|a$ s'il existe $q \in \mathbb{Z}$ tel que $a = bq$. On dit également que b est un **diviseur** de a ou que a est un **multiple** de b.

Proposition

Pour tout $n \in \mathbb{Z}$, on a les propriétés suivantes :

- si n divise a et b, alors n divise $a - b$ et $a + b$,
- si n divise a, alors n divise ka pour tout $k \in \mathbb{Z}$.

Définition

Soient $a, b \in \mathbb{Z}^*$. Alors l'ensemble des diviseurs > 0 communs à a et b admet un plus grand élément appelé **plus grand commun diviseur** de a et b. On le note : $pgcd(a, b)$.

Lemme

Soient $a, b \in \mathbb{N}^*$. Écrivons la division euclidienne $a = bq + r$. On a :

$$pgcd(a, b) = pgcd(b, r)$$

Rappels d'arithmétique
PGCD

Théorème de Bezout

Soient $a, b \in \mathbb{Z}$. Il existe des entiers $u, v \in \mathbb{Z}$ tels que :

$$au + bv = pgcd(a, b)$$

Remarque : Les coefficients u et v peuvent être calculés de manière efficace à l'aide l'algorithme d'Euclide étendu.

Définition

Soient $a, b \in \mathbb{Z}$. a et b sont dits **premiers entre eux** si $pgcd(a, b) = 1$.

entrée : $(a, b) \in \mathbb{N}^2$ avec $a > b$
sortie : $(x, y, r) \in \mathbb{N}^2 \times \mathbb{Z}$ tel que $a.x + b.y = r = pgcd(a, b)$

$x, y, r \leftarrow\ 1{,}0{,}a$
$x', y', r' \leftarrow\ 0{,}1{,}b$
répéter
  $q, r'' \leftarrow$ quotient et reste de la division euclidienne de r par r'
  $x'' \leftarrow\ x - q.x'$
  $y'' \leftarrow\ y - q.y'$
  $x, y, r \leftarrow\ x', y', r'$
  $x', y', r' \leftarrow\ x'', y'', r''$
jusqu'à $r' = 0$
retourner x, y, r

Rappels d'arithmétique

Exercice

En utilisant l'algorithme d'Euclide étendu, calculer x et y tel que :

$$14630.x + 15708.y = pgcd(14630, 15708)$$

Rappels d'arithmétique

Exercice

En utilisant l'algorithme d'Euclide étendu, calculer x et y tel que :

$$14630.x + 15708.y = pgcd(14630, 15708)$$

Ne pas oublier que a doit être le plus grand des deux nombres.

i	x	y	r
0	1	0	15708
1	0	1	14630
2	1	-1	1078
3	-13	14	616
4	14	-15	462
5	-27	29	154
6	95	-102	0

$$14630.(29) + 15708.(-27) = 154 = pgcd(14630, 15708)$$

Définition

Soit $n \in \mathbb{N}$. On dit que $a, b \in \mathbb{Z}$ sont **congrus modulo** n si n divise $a - b$. On le note $a \equiv b \pmod{n}$.

Proposition

La relation de congruence modulo n est une relation d'équivalence dans $\mathbb{Z}$. Autrement dit, modulo n, on a :

- $a \equiv a$ (réflexivité),
- $a \equiv b \Rightarrow b \equiv a$ (symétrie),
- $a \equiv b$ et $b \equiv c \Rightarrow a \equiv c$ (transitivité).

Rappels d'arithmétique

Congruences

Proposition (compatibilité avec l'addition et la multiplication)

Soit $n \in \mathbb{N}$. Supposons que l'on ait, modulo n, $a \equiv a'$ et $b \equiv b'$, alors on a :

- $a + b \equiv a' + b' \pmod{n}$,
- $ab \equiv a'b' \pmod{n}$,
- $a^k \equiv a'^k \pmod{n}$ pour tout $k \in \mathbb{N}$.

Proposition (inverse modulo n pour la multiplication)

Soient $a, n \in \mathbb{N}$ et **premiers entre eux**. Alors il existe $b > 0$ tel que $ab \equiv 1 \pmod{n}$.

Rappels d'arithmétique

Inverse modulo n pour la multiplication

Si b est l'inverse modulaire de a pour la multiplication modulo n, alors

$$ab \equiv 1 \pmod{n}$$

Donc il existe un entier v tel que :

$$1 - ab = nv$$

ou encore, tel que :

$$ab + nv = 1$$

Puisque a et n sont premiers entre eux, on peut trouver b avec l'algorithme d'Euclide étendu.

Théorème : Petit théorème de Fermat

Si p est un nombre premier et $a \in \mathbb{Z}$ alors

$$a^p \equiv a \pmod{p}$$

De plus, si p ne divise pas a alors

$$a^{p-1} \equiv 1 \pmod{p}$$

Théorème : Petit théorème de Fermat amélioré

Soient p et q deux nombres premiers distincts et soit $n = pq$. Pour tout $a \in \mathbb{Z}$ tel que $pgcd(a, n) = 1$, on a :

$$a^{(p-1)(q-1)} \equiv 1 \pmod{n}$$

- On note $\varphi(n) = (p-1)(q-1)$ l'**indicatrice d'Euler**.
- L'hypothèse $pgcd(a, n) = 1$ équivaut à ce que a ne soit divisible ni par p, ni par q.

RSA : Chiffrement asymétrique

RSA

- Le système RSA, du nom de ses concepteurs *Rivest*, *Shamir* et *Adleman* est le premier système de chiffrement à clé publique robuste à avoir été inventé (en 1977).
- Il est toujours largement utilisé lorque l'on veut échanger des données de manière sécurisée sur Internet.
- La robustesse du système RSA repose sur le fait que l'on ne sait pas, avec les moyens et savoirs actuels, obtenir la clé privée à partir de la simple connaissance de la clé publique.

Le principe de RSA

Création des clés

- choisir deux (grands) nombres premiers distincts p et q ;
- calculer leur produit $n = pq$, appelé **module de chiffrement** ;
- calculer l'indicatrice d'Euler en n : $\varphi(n) = (p-1)(q-1)$;
- choisir un entier naturel e, appelé **exposant de chiffrement**, tel que $3 < e < \varphi(n)$ et $pgcd(e, \varphi(n)) = 1$
- calculer l'entier naturel d, inverse modulaire de e pour la multiplication modulo $\varphi(n)$ et strictement inférieur à $\varphi(n)$, appelé **exposant de déchiffrement** : $ed \equiv 1 \ (mod\ \varphi(n))$.
- La **clé publique** est constituée par le coupe (e, n).
- La **clé privée** est le nombre d (ou le couple (d, n)).

Pour chiffrer un texte clair $\mathbf{m} \in \mathbb{N}^*$ et déchiffrer un chiffré $\mathbf{c} \in \mathbb{N}^*$:

$\boxed{Enc_{\mathbf{k}_{pub}}(\mathbf{m}) = \mathbf{m}^e \mod n}$ et $\boxed{Dec_{\mathbf{k}_{priv}}(\mathbf{c}) = \mathbf{c}^d \mod n}$

Exemple (1)

- Alice choisit deux nombres premiers distincts : $p = 5$ et $q = 17$.
- Elle peut calculer : $n = pq = 5 \times 17 = 85$ et $\varphi(n) = (p-1)(q-1) = 4 \times 16 = 64$.
- Elle choisit un exposant de chiffrement $e = 5$, et on a bien $pgcd(e, \varphi(n)) = pgcd(5, 64) = 1$.
- Elle calcule l'exposant de déchiffrement : $d = 13$.
- Sa clé publique est donc $(5, 85)$, et sa clé privée est 13.

Exemple (2)

- Si Bob veut lui envoyer le message $m = 10$
- Il récupère la clé publique d'Alice ($e = 5, n = 85$) et calcule $x \equiv 10^5 (mod\ 85)$. Le message chiffré est donc $x = 40$, qu'il envoit à Alice.
- Avec sa clé privée, Alice peut déchiffrer le message de Bob en calculant $m \equiv 40^{13} (mod\ 85)$. Elle retrouve bien le message $m = 10$.

L'exponentiation modulaire

En cryptographie asymétrique, en particulier avec l'algorithme RSA, on a souvent besoin de calculer $b^e (mod\ m)$ explicitement où b, e, m sont de "grands" entiers naturels. Ce calcul s'appelle **exponentiation modulaire** : b est appelé la **base**, e l'**exposant** et m le **module**.

Par exemple, considérons le calcul de $341^{943} (mod\ 1403)$. La méthode naïve consisterait à élever 341 à la puissance 943 puis effectuer la division euclidienne du résultat par 1403, le reste de cette division étant le nombre cherché.

Une première difficulté est d'avoir à effectuer 942 multiplications (en pratique, les nombres sont beaucoup plus grands) suivies d'une division euclidienne.

Mais la difficulté principale (rédhibitoire en pratique) est, qu'avec cette méthode, on est conduit à manipuler des nombres de plus en plus grands.

L'exponentiation rapide

On veut calculer $341^{943} (mod\ 1403)$.

On vient calculer les puissances de 341 modulo 1403 suivantes :

$$
\begin{aligned}
341^{1} &\equiv 341 \mod 1403\\
341^{2} &\equiv 1235 \mod 1403\\
341^{4} &\equiv 164 \mod 1403\\
341^{8} &\equiv 239 \mod 1403\\
341^{16} &\equiv 1001 \mod 1403\\
341^{32} &\equiv 259 \mod 1403\\
341^{64} &\equiv 1140 \mod 1403\\
341^{128} &\equiv 422 \mod 1403\\
341^{256} &\equiv 1306 \mod 1403\\
341^{512} &\equiv 991 \mod 1403
\end{aligned}
$$

L'exponentiation rapide

$943 = 2^0+2^1+2^2+2^3+2^5+2^7+2^8+2^9 = 1+2+4+8+32+128+256+512$
Donc :

$$
\begin{aligned}
341^{943} &\equiv 341^{(1+2+4+8+32+128+256+512)} && \mod 1403 \\
&\equiv 341^1 \times 341^2 \times 341^4 \times 341^8 \times \\
&\quad 341^{32} \times 341^{128} \times 341^{256} \times 341^{512} && \mod 1403 \\
&\equiv 341 \times 1235 \times 164 \times 239 \\
&\quad \times 259 \times 422 \times 1306 \times 991 && \mod 1403 \\
&\equiv 980 && \mod 1403
\end{aligned}
$$

L'exponentiation par carrés

Une autre manière de voir l'exponentiation rapide, c'est avec l'algorithme d'**exponentiation par carrés** qui combine deux opérations élémentaires : l'élévation au carré et la multiplication par la base b.

On commence par décomposer l'exposant e de la manière suivante : si e est pair, on le divise par 2, sinon on lui retranche 1 et on le divise par 2 et ensuite on recommence avec le quotient jusqu'à ce qu'on arrive (nécessairement) à 1. On écrit dans l'ordre inverse d'obtention cette suite de nombres. Dans notre exemple avec e = 943, cela donne la suite : 1, 2, 3, 6, 7, 14, 28, 29, 58, 116, 117, 234, 235, 470, 471, 942, 943.

Pour $b = 341$, on a, modulo 1403 : $b^1 \equiv 341$, $b^2 \equiv 1235$, $b^3 \equiv 235$, $b^6 \equiv 508$, $b^7 \equiv 659$, $b^{14} \equiv 754$, $b^{28} \equiv 301$, $b^{29} \equiv 222$, $b^{58} \equiv 179$, $b^{116} \equiv 1175$, $b^{117} \equiv 820$, $b^{234} \equiv 363$, $b^{235} \equiv 319$, $b^{470} \equiv 745$, $b^{471} \equiv 102$, $b^{942} \equiv 583$ et $b^{943} \equiv 980$.
Le calcul a nécessité seulement 16 multiplications au lieu de 942 avec la méthode naïve !

Utilisation de RSA en pratique

- Comme tout système à clé publique, le chiffrement RSA est **coûteux** en temps calcul. Plus précisément, il est (beaucoup) plus coûteux qu'un chiffrement *symétrique* de robustesse équivalente comme AES par exemple.
- C'est pour cela, que souvent en pratique, on ne l'utilise que pour transmettre la clé secrète d'un chiffrement symétrique.
- Concrètement supposons qu'Alice veuille échanger des données de manière sécurisée avec Bob. Elle va ainsi procéder :
  - Elle va tout d'abord générer une clé secrète pour le chiffrement symétrique dont le choix (public) a été fait au préalable avec son correspondant.
  - Elle chiffre cette clé secrète avec la clé publique (RSA) de Bob et transmet la clé ansi chiffrée à ce dernier.
  - Bob déchiffre la clé secrète avec sa clé privée (RSA).
  - Les deux correspondants étant maintenant en possession de la clé secrète, le chiffrement symétrique des données peut commencer.

Signature numérique

Signature numérique avec RSA

- Le système RSA permet également de *signer* des données numériques.
- Pour ce faire, il suffit d'inverser les rôles de e et d dans l'algorithme décrit précédemment.
- Supposons qu'Alice veuille envoyer à Bob un message M signé. Avec sa clé privée d, elle va générer la signature $S = M^d (mod\ n)$ du message M, où n est la clé publique.
- Elle envoie ensuite à Bob le couple (M, S).
- Pour authentifier la signature, Bob, lorsqu'il reçoit le couple (M, S), doit d'abord calculer $M' = S^e (mod\ n)$, où e est l'exposant public de la clé d'Alice, et ensuite vérifier que $M' = M$.

Signature numérique

Alice

Bob, ...

génère $\mathbf{k}_{pub}$ et $\mathbf{k}_{priv}$

$\xrightarrow{\mathbf{k}_{pub}}$ $\mathbf{k}_{pub}$

$\mathbf{m}$: message

$\sigma = Signature(\mathbf{k}_{priv}, \mathbf{m})$ $\xrightarrow{\mathbf{m}, \sigma}$ $\tilde{\mathbf{m}}, \tilde{\sigma}$

peut être différent de $\mathbf{m}$ et σ si Eve intercepte et modifie les messages.

$Verifie(\mathbf{k}_{pub}, \tilde{\mathbf{m}}, \tilde{\sigma})$

Signature vs. MAC

Quelle est la différence entre une signature numérique et un code d'authentification de message ?

- Les deux répondent à la question d'**intégrité** mais pas de confidentialité.
- L'un utilise un système symétrique, tandis que l'autre utilise un système asymétrique. MAC a besoin de partager une clé secrète, ce qui n'est pas le cas d'une signature numérique.
- Dans le cas d'un MAC, l'émetteur et le récepteur partagent la clé secrète, donc tous les deux peuvent signer. C'est pour cela que MAC ne répond pas à la question de **non répudiation**, à l'opposé des signatures numériques.

Signature RSA

Il y a des attaques par falsification triviales ou des attaques de maléabilité qui existent sur la signature RSA.

Pour résoudre cela, on applique une fonction de hachage cryptographique avant de signer : **"Hash & Sign"**.

Hash & Sign

Alice

Bob, ...

génère $\mathbf{k}_{pub}$ et $\mathbf{k}_{priv}$ $\xrightarrow{\mathbf{k}_{pub}}$ $\mathbf{k}_{pub}$

$\mathbf{m}$: message

$H(\mathbf{m})$: hash

$\sigma = Signature(\mathbf{k}_{priv}, H(\mathbf{m}))$ $\xrightarrow{\mathbf{m}, \sigma}$ $\tilde{\mathbf{m}}, \tilde{\sigma}$

peut être différent de $\mathbf{m}$ et σ si Eve intercepte et modifie les messages.

$Verifie(\mathbf{k}_{pub}, H(\tilde{\mathbf{m}}), \tilde{\sigma})$

Certificats

Position du problème

- En cryptographie *asymétrique*, chaque acteur dispose d'un couple de clés : une clé **publique** et une clé *privée*.
- La clé privée, comme son nom l'indique, ne doit être à disposition que de son propriétaire (qui en a l'entière responsabilité). En revanche, la clé publique **doit être accessible** à tout un chacun.
- La diffusion de cette clé publique doit respecter plusieurs critères :
  - *authentification* : on doit être sûr que la clé est bien celle de la personne avec qui on va échanger des données confidentielles (risque de l'attaque "man in the middle").
  - *confiance* : la personne en question doit être digne de confiance.
  - *validité* : une clé publique a une durée de vie en général finie. On doit donc pouvoir vérifier cette dernière.
- Un des moyens de résoudre ce problème : les *certificats* et *autorités de certification* gérés au sein d'une **infrastructure de gestion de clés** - IGC (Public Key Infrastructure - PKI).

Certificats : pourquoi ?

- Le chiffrement et la signature supposent l'authenticité des clés publiques, disponible sur un annuaire ou un serveur web.
  - La signature garantie que le message provient bien du détenteur de la clé privée. Mais : à qui appartient la clé privée / publique ?
  - Le chiffrement garantie que le message ne pourra être déchiffré que par le détenteur de la clé privée (associé à la clé publique utilisée lors du chiffrement). Mais : à qui appartient cette clé publique ?
- Est-ce qu'on est sûre qu'il ne s'agit pas d'un usurpateur ?

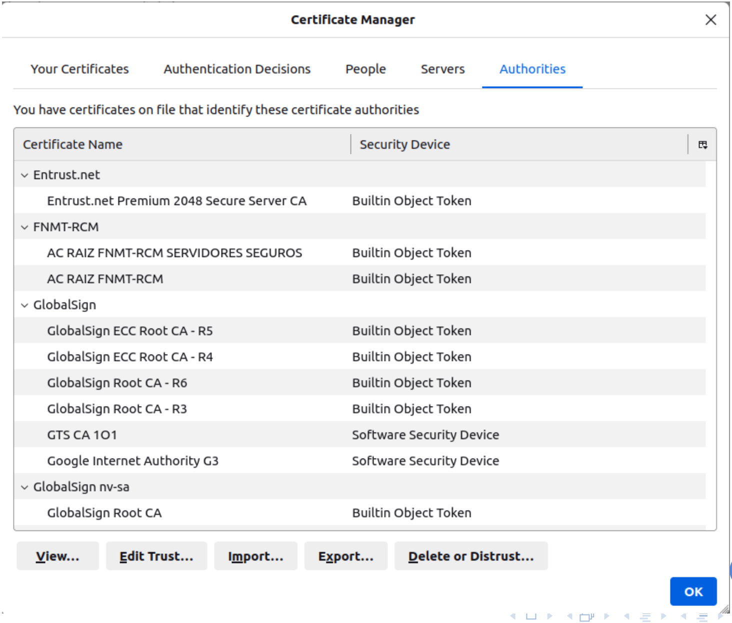
Qu'est ce qu'un certificat ?

- Formellement, c'est un document numérique contenant une **clé publique** ainsi que d'autres informations associées à cette clé.
- Ce document peut être authentifié (signature numérique) de différentes manières :
  - soit par une autorité de certification (AC). C'est la norme X.509 de l'ISO définie en 1988 dans le cadre du projet X.500. Le modèle sous-jacent est un système hiérarchique d'autorités de certification.
  - soit par quiconque appartenant à un réseau de confiance. C'est par exemple la toile de confiance (web of trust) d'OpenPGP.
- Dans le modèle X.509, une des limites est la **confiance** que l'on peut accorder à une AC. Cette confiance est essentielle car elle conditionne tout le reste. En effet si une AC est compromise, l'ensemble des clés qui en dépendent est compromis.
- C'est ce problème de confiance qui a amené Paul Zimmermann, auteur de PGP (Pretty Good Privacy), à introduire, au début des années 90, le concept de *toile de confiance*.

Qu'est ce qu'un certificat ?

- Le certificat vise à effectuer le lien entre une identité (d'une personne) et une bi-clé (privée / publique)
  - Il est délivré par une autorité de certification
  - Il est nominatif
  - Il est destiné à un usage unique (signature ou chiffrement)
  - Il a une durée de validité donnée
  - Il est révocable

Quelques autorités de certification (navigateur Firefox)



Certificat X.509

Données	`Version:`	Version du type de certificat X.509
	`Serial number:`	Numéro de série au sein de l'AC
	`Signature algorithm:`	Algorithme de signature utilisé
	`Issuer:`	Identité de l'AC (*Distinguished Name*) qui a émis ce certificat

	`Validity`
	`Not before: xx`
	`Not after : xx`
		Période de validité du certificat : dates de début et de fin

	`Subject:`	Identité du propriétaire (*Distinguished Name*) du certificat
	`Subject Public Key Info:` `Public Key Algorithm:`	Informations sur la clé publique et les paramètres de celle-ci

	`X509v3 extensions:`
	`Extension name:`
	`Extension value`
	`...`
		Extensions optionnelles propres à la version 3

Signature	`Signature algorithm:`	Algorithme utilisé pour la signature
	`Signature`	Signature du certificat

Quelques extensions X.509v3

`X509v3 Basic Constraints:`	Indique s'il s'agit du certificat d'une AC ou non
`X509v3 Key Usage:`	Précise les fonctionnalités du certificat. Par exemple, il peut être utilisé pour signer d'autres certificats (*Certificate sign*)
`X509v3 subjectAltName:`	Autres noms du propriétaire du certificat. Ce sont des alias du champ *Subject*
`X509v3 issuerAltName:`	Autres noms de l'émetteur du certificat. Ce sont des alias du champ *Issuer*
`X509v3 CRL Distribution points:`	URI de la *Certificat Revocation List (CRL)* permettant de connaître le statut du certificat

Certificats : vérification

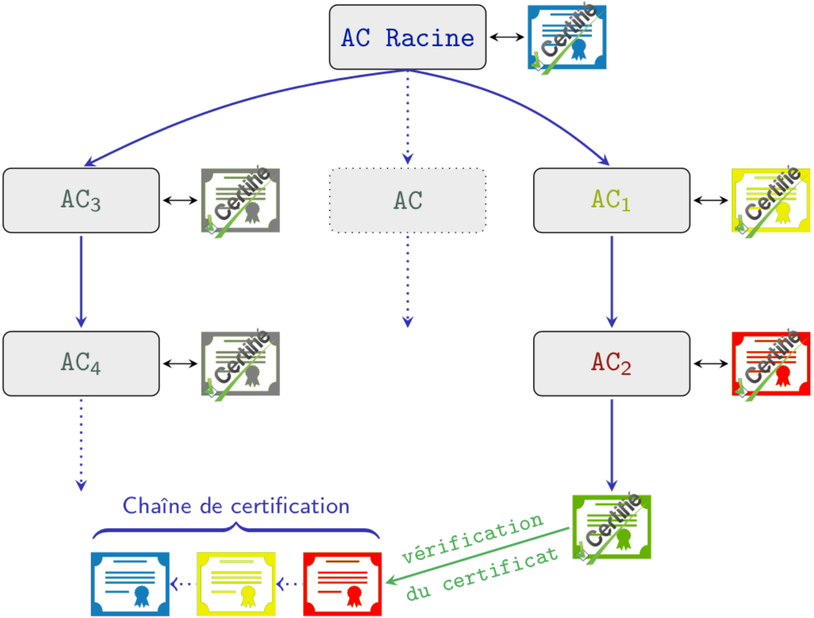
- Chiffrer un message nécessite la clé publique du destinataire.
- Vérifier la signature nécessite la clé publique de l'émetteur.
- Au lieu de demander la clé publique au destinataire/émetteur, on va **demander le certificat** (trouvable dans l'annuaire), **vérifier** le certificat (vérifier la signature du certificat) et en **extraire la clé publique**.

Certificats : utilisation

- Vérification de la **signature** :
  Signature $=$ Chiffrement(Hash(Données))$_{KprivAC}$
- Vérification de la **validité** : date de début / date de fin
- Vérification de la **révocation** : a-t-il été révoqué ?
- Vérification de l'**usage** : l'algorithme avec lequel je peux utiliser le certificat

Autorité de certification : principe

- Pour être **valide**, un certificat doit être **signé par une AC**.
- Une AC possède son propre couple (clé privée, clé publique) associé à un certificat qui peut être, soit **auto-signé**, soit **signé par une autre AC**.
- N'importe qui peut se déclarer AC, ce qui pose évidemment le problème de la **confiance** en une AC. Or cette confiance est cruciale car la securité de ce système hiérarchique repose, pour une grande part, sur celle-ci.
- La confiance accordée à une AC est **héritée** par toutes les ACs filles. Autrement dit, faire confiance à une AC *racine*, implique que l'on accorde la même confiance à toutes les ACs qui en dérivent.
- Deux ACs peuvent s'entendre afin de signer chacune le certificat de l'autre : c'est la notion de **confiance croisée**.
- L'AC est **garante** des informations contenues dans les certificats qu'elle délivre.


AC Racine
Certifié
AC3
AC
AC1
AC4
AC2
Chaîne de certification
vérification du certificat

Structure d'une infrastructure de gestion de clés

- Une définition formelle : une IGC (*Public Key Infrastructure - PKI*) est un ensemble de personnes, matériels et logiciels, régis par des règles et des procédures et permettant de **créer**, **gérer** et **distribuer** des certificats X.509.
- C'est donc à la fois une entité **administrative** et une entité **technique** qui aura en charge l'ensemble des procédures concernant les certificats dont elle a la responsabilité.
- Plus précisément, une IGC assure les missions suivantes :
  - création et révocation des certificats X.509 ;
  - diffusion et publication des certificats X.509 (via un annuaire LDAP par exemple) ;
  - plus rarement, un service de séquestre et de recouvrement des clés privées. Ce service est bien sûr très utile en cas de perte de la clé privée de votre certificat. Le problème est que votre clé privée perd toute légitimité car vous la partagez avec un tiers.

Les éléments constitutifs d'une IGC

- Autorité d'Enregistrement (AE) : elle reçoit et traite les demandes de création, renouvellement et révocation de certificats. Elle doit notamment s'assurer de l'identité des demandeurs.
- Opérateur de Certification (OC) : il effectue toutes les opérations demandées par l'AC nécessitant la clé privée de celle-ci. Il n'est en principe pas connecté au réseau.
- Service de Publication (SP) : il met à disposition de tous, via un annuaire (le plus souvent LDAP), les certificats issus de l'IGC, le certificat de l'AC et éventuellement les listes de révocation (CRL).
- Service de Validation (SV) : il permet à tout utilisateur de vérifier la validité d'un certificat (expiration, vol/perte de la clé privée associée,...).
- Service de Séquestre (SS) : il stocke au sein de l'IGC les couples (clé privée, clé publique) des certificats produits. Dangereux !

Demande de certificat

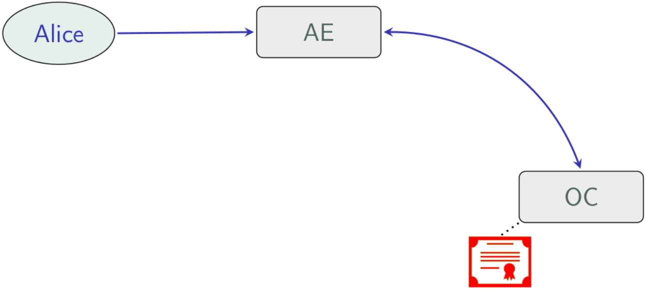
Alice

Alice souhaite obtenir un certificat

Demande de certificat

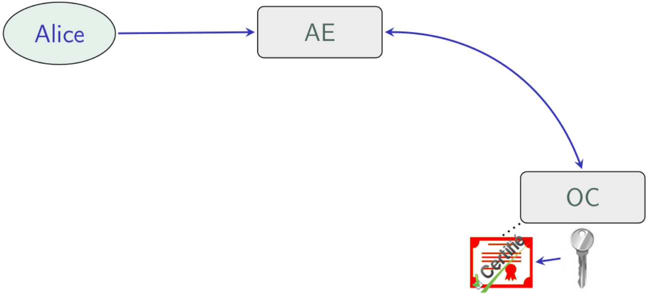
Alice envoie sa requête à l'AE

Demande de certificat



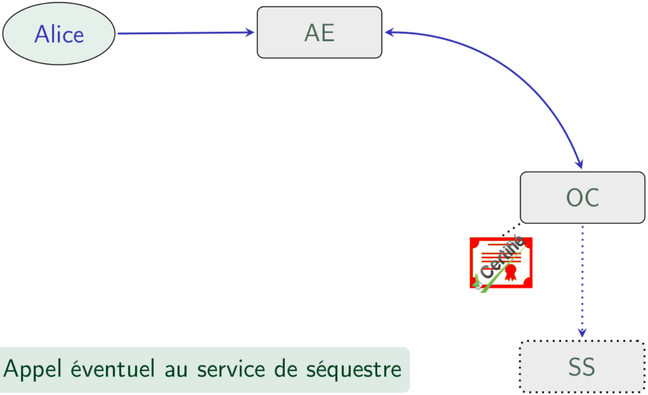
L'AE vérifie l'identité d'Alice et transmet à l'OC

Demande de certificat



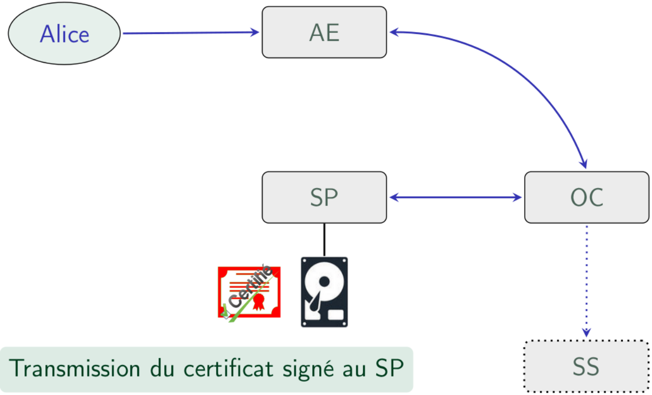
L'OC signe le certificat avec la clé privée de l'AC

Demande de certificat

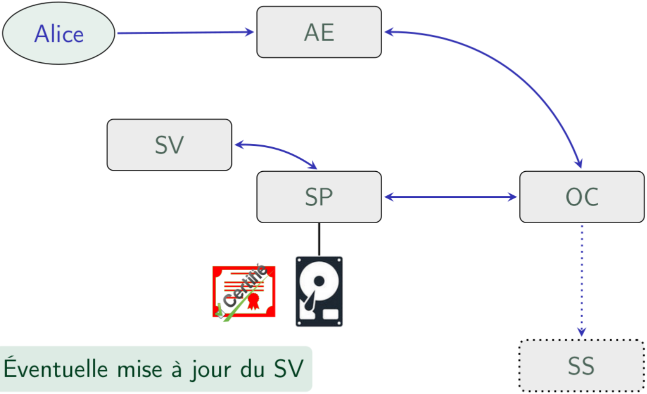


Appel éventuel au service de séquestre

Demande de certificat

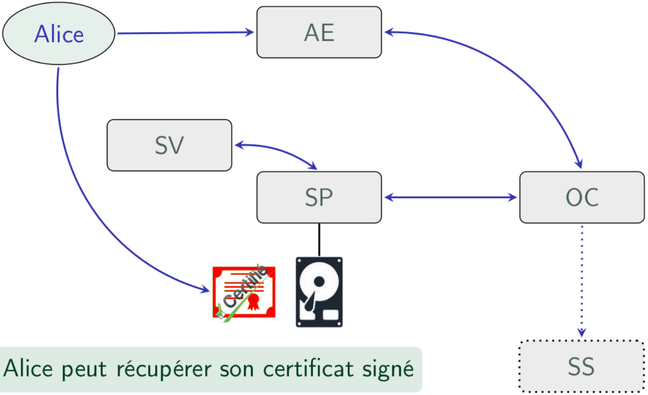


Demande de certificat



Éventuelle mise à jour du SV

Demande de certificat



Alice peut récupérer son certificat signé