Chapitre 1

Introduction à la Sécurité

1.1 Objectifs de la sécurité

La sécurité vise à assurer plusieurs propriétés :

La confidentialité : c'est la propriété qui garantit que les informations transmises ne sont compréhensibles que par les entités autorisées.

L'authentification : c'est la propriété qui consiste à vérifier l'identité d'un utilisateur avant de lui donner l'accès à une ressource.

L'intégrité : c'est la propriété qui consiste à vérifier si les informations n'ont pas été modifiées durant la transmission.

La disponibilité : c'est la propriété qui permet de garantir l'accès aux données.

La non-répudiation : c'est la propriété qui permet d'avoir une preuve comme quoi un utilisateur a envoyé (ou reçu) un message particulier. Cette propriété permet d'empêcher l'utilisateur de nier l'envoie (ou réception) du message en question.

1.2 Les scénarios d'attaques

1.2.1 Attaque passive

Dans ce genre d'attaques, les informations ne sont pas modifiées. L'attaquant collecte seulement les informations qui circulent sur le réseau.

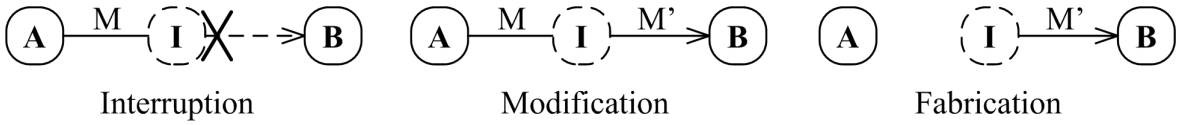
1.2.2 Attaque active

Il y a trois cas possibles pour mener une attaque active :

L'interruption : l'intrus intercepte le message envoyé par l'utilisateur $\mathcal{A}$ pour $\mathcal{B}$ et l'interrompt.

La modification : l'intrus intercepte le message envoyé par l'utilisateur $\mathcal{A}$ et le modifie avant de le faire suivre à l'utilisateur $\mathcal{B}$.

La fabrication : L'intrus fabrique un message et l'envoie à l'utilisateur $\mathcal{B}$ en se passant pour l'utilisateur $\mathcal{A}$.



1.3 Concepts de base sur la cryptographie

Chiffrement et déchiffrement : consiste à transformer une donnée afin de la rendre incompréhensible par un utilisateur autre que celui qui l'a créée et celui qui va la recevoir.

Chiffré : c'est le résultat de chiffrement d'une donnée.

Clé : il s'agit d'un paramètre impliqué dans les opérations de chiffrement et de déchiffrement et qui est partagé entre l'émetteur et le récepteur.

Cryptanalyse : elle a pour but de retrouver la donnée en claire à partir d'un ou plusieurs chiffrés sans connaitre les clés et/ou l'algorithme de chiffrement.

Canal : moyen de transport de l'information.

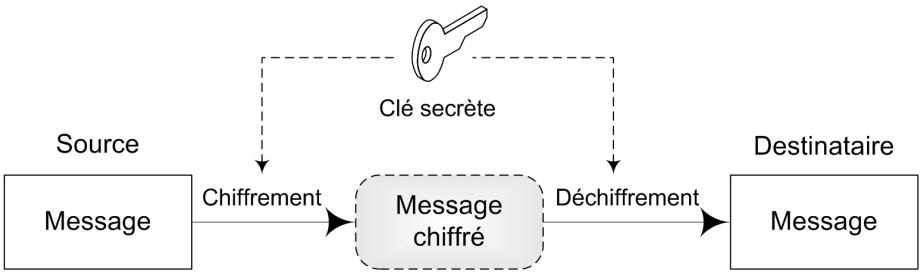
Canal sécurisé : canal où l'intrus n'a pas la possibilité d'altérer les messages.

Canal sécuritaire : canal qui n'est pas physiquement accessible à l'intrus.

1.4 Les algorithmes cryptographiques

1.4.1 Algorithmes à clés symétriques

Dans ce type d'algorithme, la même clé est utilisée à la fois pour le chiffrement et le déchiffrement (DES, AES, IDEA, ... etc.).

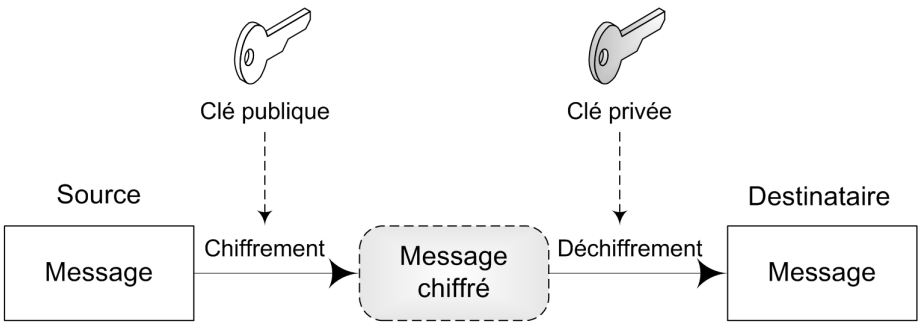


1.4.1.1 Exemple

- ⋆ Le message $\mathcal{M} = 139$.
- ⋆ L'opérateur de chiffrement et de déchiffrement : $\oplus$ (XOR).
- ⋆ La clé $\mathcal{K} = 199$.
- ⋆ $\mathcal{C} = \mathcal{M} \oplus \mathcal{K} = 139 \oplus 199 = 76$.
- ⋆ $\mathcal{M} = \mathcal{C} \oplus \mathcal{K} = 76 \oplus 199 = 139$.

1.4.2 Algorithmes à clés asymétriques

Dans ce type d'algorithmes, chaque entité possède une paire de clés : *publique* et *privée*. La clé publique est utilisée pour le chiffrement et la clé privée pour le déchiffrement (RSA, Elgamal, Rabin, Merkel-Hellman, ... etc.).



La signature numérique consiste à générer un condensé à partir d'un message et le chiffrer en utilisant la clé privée de l'émetteur. Ce dernier, ensuite, envoie le message en clair avec la signature. Pour vérifier la validité de cette signature, le récepteur recalcule le condensé à partir du message en clair et déchiffre la signature. Ensuite, il vérifie l'égalité des deux résultats.

1.4.3 Fonctions de hachage

Il s'agit de la troisième famille d'algorithmes cryptographiques. Le principe est qu'un message $\mathcal{M}$ de longueur quelconque est transformé en une valeur h de longueur fixe et inférieure à celle du départ ($h = H(\mathcal{M})$) et qui représente d'une manière unique le message $\mathcal{M}$. Deux caractéristiques importantes sont les suivantes :

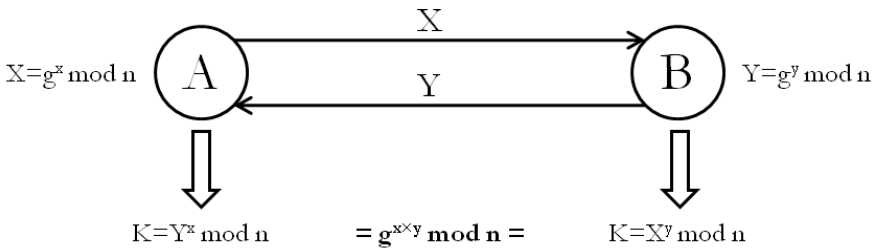
1. A partir de h, il est impossible de retrouver $\mathcal{M}$.
2. Etant donné $h = H(\mathcal{M})$, il est impossible de trouver $\mathcal{M}'$ tel que $H(\mathcal{M}') = h$.

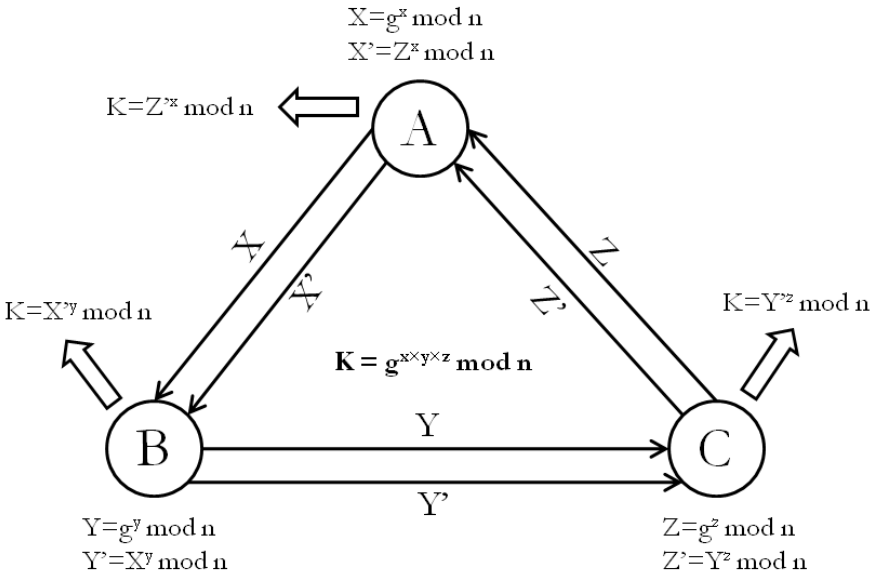
Chapitre 2

Algorithmes de Chiffrement à Clés Publiques

2.1 Protoccole de Diffie-Hellman

Le protocole d'échange de Diffie-Hellman est une méthode par laquelle deux utilisateurs peuvent partager une clé de chiffrement symétrique. Le protocole tire sa sécurité du problème du Logarithme discret. Il peut s'exécuter avec deux ou plusieurs utilisateurs.





2.2 Chiffrement de RSA

L'algorithme de chiffrement de RSA tire sa sécurité du problème de factorisation des grands nombres entiers. L'algorithme se déroule en trois étapes :

1. *Création de clés :* On choisit deux nombre premiers p et q et on calcule n et $\phi(n)$ tels que $n = p \times q$ et $\phi(n) = (p-1) \times (q-1)$. Ensuite, on choisit $e < \phi(n)$ tel que $pgcd(e, \phi(n)) = 1$, et enfin on calcule d tel que $e \times d \bmod \phi(n) = 1$. La clé publique est (e, n) et la clé privée est (d, n).
2. *Chiffrement :* Le message à chiffrer doit être strictement inferieur à n. Pour chiffrer un message $\mathcal{M}$, on calcule $\mathcal{C} = \mathcal{M}^e \bmod n$.
3. *Déchiffrement :* Pour déchiffrer le message, on calcule $\mathcal{M} = \mathcal{C}^d \bmod n$.

2.2.1 Exemple

⋆ $p = 19$, $q = 23 \Rightarrow n = 19 \times 23 = 437$ et $\phi(n) = (19-1) \times (23-1) = 396$.

⋆ $e = 17 \Rightarrow 17d \bmod 396 = 1 \Rightarrow d = 233$.

⋆ Pour $\mathcal{M} = 351 \Rightarrow \mathcal{C} = 351^{17} \bmod 437 = 150$.

⋆ $\mathcal{M} = 150^{233} \bmod 437 = 351$.

2.3 Chiffrement de Rabin

L'algorithme de chiffrement de *Rabin* tire sa sécurité du problème de factorisation des grands nombres entiers. L'algorithme se déroule en trois étapes :

1. *Création de clés :* On choisit deux nombres premiers p et q, tels que $p \bmod 4 = 3$ et $q \bmod 4 = 3$. La clé publique est $n = p \times q$ et la clé privée est (p, q).
2. *Chiffrement :* Le message à chiffrer doit être strictement inférieur à n. Pour chiffrer un message $\mathcal{M}$, on calcule $\mathcal{C} = \mathcal{M}^2 \bmod n$.
3. *Déchiffrement :* Pour déchiffrer le message, on calcule tout d'abord m_p, m_q, y_p, et y_q tels que $m_p = \mathcal{C}^{(p+1)/4} \bmod p$, $m_q = \mathcal{C}^{(q+1)/4} \bmod q$, et $p \times y_p + q \times y_q = 1$. Ensuite, on calcule $\mathcal{R}$ et $\mathcal{S}$, tels que $\mathcal{R} = (p \times m_q \times y_p + q \times m_p \times y_q) \bmod n$, et $\mathcal{S} = (p \times m_q \times y_p - q \times m_p \times y_q) \bmod n$. L'une des solutions $\{\mathcal{R}, \mathcal{S}, n-\mathcal{R}, n-\mathcal{S}\}$ est le message $\mathcal{M}$.

2.3.1 Exemple

- ⋆ $p = 11$, $q = 23 \Rightarrow n = 253$.
- ⋆ Pour $\mathcal{M} = 158 \Rightarrow \mathcal{C} = 158^2 \bmod 253 = 170$.
- ⋆ $m_p = 170^{(11+1)/4} \bmod 11 = 4$.
- ⋆ $m_q = 170^{(23+1)/4} \bmod 23 = 3$.
- ⋆ $11y_p + 23y_p = 1 \Rightarrow y_p = -2$ et $y_q = 1$.
- ⋆ $\mathcal{R} = (11 \times 3 \times (-2) + 23 \times 4 \times 1) \bmod 253 = 26$.
- ⋆ $\mathcal{S} = (11 \times 3 \times (-2) - 23 \times 4 \times 1) \bmod 253 = 95$.
- ⋆ Les solutions possibles sont : $\{26, 95, (253 - 26) = 227, (253 - 95) = \underline{\mathbf{158}}\}$.

2.4 Chiffrement de Merkel-Hellman

L'algorithme de chiffrement de *Merkel-Hellman* tire sa sécurité du problème de *sac-à-dos*. Etant données des valeurs entières $\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_k$ et un poids $\mathcal{T}$. Est-il possible de déterminer des valeurs binaires $b_1, b_2, ..., b_k$ de telle sorte que $\mathcal{T} = b_1\mathcal{P}_1 + b_2\mathcal{P}_2 + ... + b_k\mathcal{P}_k$? Si la suite des poids $\mathcal{P}_i$ est super-croissante (chaque poids $\mathcal{P}_i$ est strictement supérieur à la somme de tous les poids précédents), alors il existe un algorithme polynomial permettant de déterminer les valeurs de b_i :

Pour i=k **à** 1 **Faire**
  Si $T \geq P_i$ **Alors** $T \leftarrow T - P_i$;
    $b_i \leftarrow 1$;
  Sinon $b_i \leftarrow 0$;
Fin pour
Si $T = 0$ **Alors** $\{b_1, b_2, \ldots, b_k\}$ est la solution ;
Sinon Il n'existe pas de solutions ;

On peut vérifier qu'avec la suite super-croissante $\{2, 3, 6, 12\}$ pour $\mathcal{T} = 15$ on obtient la solution $\{0, 1, 0, 1\}$.

La résolution d'une suite non super-croissante est un problème NP-complet. La sécurité du système de chiffrement *Merkle-Hellman* est basé sur cette difficulté. Il se déroule en trois étapes :

1. *Création de clés :* On choisit une suite super-croissante $\mathcal{A} = \{\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_k\}$.

Ensuite, on choisit deux nombres n et m tel que m est supérieur à la somme de tous les $\mathcal{P}_i$, et n n'a de facteur commun avec aucun nombre de la suite. On calcule, ensuite, la suite non super-croissante $\mathcal{B} = \{\mathcal{P}'_1, \mathcal{P}'_2, ..., \mathcal{P}'_k\}$ tel que $\mathcal{P}'_i = n\mathcal{P}_i \bmod m$. La clé publique est $\mathcal{B}$, et la clé privée est $(\mathcal{A}, n, m)$.

2. *Chiffrement :* Le message à chiffrer doit être une suite binaire tel que $\mathcal{M} = b_1 b_2 ... b_k$. On calcule $\mathcal{C} = b_1\mathcal{P}'_1 + b_2\mathcal{P}'_2 + ... + b_k\mathcal{P}'_k$.
3. *Déchiffrement :* Pour déchiffrer le message, on calcule tout d'abord n^{-1} tel que $n \times n^{-1} \bmod m = 1$. Ensuite, on calcule $\mathcal{T} = n^{-1} \times \mathcal{C} \bmod m$. Enfin, on calcule les valeurs b_i du message $\mathcal{M}$ en utilisant l'algorithme de résolution d'une suite super-croissante.

2.4.1 Exemple

- ⋆ $\mathcal{A} = \{2, 3, 6, 13, 27, 52\}$, $n = 31$ et $m = 105 \Rightarrow \mathcal{B} = \{62, 93, 81, 88, 102, 37\}$.
- ⋆ Pour $\mathcal{M} = 53 = 110101_{(2)} \Rightarrow \mathcal{C} = 1{\times}62{+}1{\times}93{+}0{\times}81{+}1{\times}88{+}0{\times}102{+}1{\times}37 = 280$.
- ⋆ $31 \times n^{-1} \bmod 105 = 1 \Rightarrow n^{-1} = 61$.
- ⋆ $\mathcal{T} = (61 \times 280) \bmod 105 = 70 = 1 \times 2 + 1 \times 3 + 0 \times 6 + 1 \times 13 + 0 \times 27 + 1 \times 52$ $\Rightarrow \mathcal{M} = 110101_{(2)} = 53$.

2.5 Chiffrement d'Elgamal

L'algorithme de chiffrement d'*Elgamal* tire sa sécurité du problème du Logarithme discret qui est un problème NP-complet. Il se déroule en trois étapes :

1. *Création de clés :* On choisit un nombre premier p et deux nombres a et g inferieur à p. Ensuite, on calcule $\mathcal{A} = g^a \bmod p$. La clé publique est $(\mathcal{A}, g, p)$ et la clé privée est a.
2. *Chiffrement :* Le message à chiffrer doit être strictement inferieur à p. Pour chiffrer le message $\mathcal{M}$, on choisit un nombre aléatoire b strictement inferieur à p et premier avec $p - 1$. Ensuite, on calcule $\mathcal{B}$ et $\mathcal{C}$, tels que $\mathcal{B} = g^b \bmod p$ et $\mathcal{C} = \mathcal{M} \times \mathcal{A}^b \bmod p$. Le message chiffré est $(\mathcal{B}, \mathcal{C})$.

3. *Déchiffrement :* Pour déchiffrer le message, on calcule $\mathcal{M} = \mathcal{C} \times \mathcal{B}^{(p-a-1)} \bmod p$.

2.5.1 Exemple

- $p = 23$, $g = 7$, $a = 6 \Rightarrow \mathcal{A} = 7^6 \bmod 23 = 4$.
- Pour $\mathcal{M} = 7$ et $b = 3 \Rightarrow \mathcal{B} = 7^3 \bmod 23 = 21$ et $\mathcal{C} = (7 \times 4^3) \bmod 23 = 11$.
- $\mathcal{M} = (11 \times 21^{(23-6-1)}) \bmod 23 = 7$.

2.5.2 Signature numérique

Pour calculer la signature numérique d'un message $\mathcal{M}$, on choisit un nombre aléatoire b strictement inférieur à p et premier avec $p - 1$. Ensuite, on calcule $\mathcal{B} = g^b \bmod p$ et $\mathcal{C}$ tel que $\mathcal{M} = (a \times \mathcal{B} + b \times \mathcal{C}) \bmod (p - 1)$. Pour vérifier la validité de la signature $(\mathcal{M}, \mathcal{B}, \mathcal{C})$, on vérifie l'égalité $\mathcal{A}^{\mathcal{B}} \times \mathcal{B}^{\mathcal{C}} \bmod p = g^{\mathcal{M}} \bmod p$.

2.5.2.1 Exemple

- $p = 23$, $g = 7$, $a = 6$, $\mathcal{A} = 4$, $\mathcal{M} = 7$, $b = 3$.
- $\mathcal{B} = 7^3 \bmod 23 = 21$.
- $\mathcal{C}$ tel que $(6 \times 21 + 3\mathcal{C}) \bmod 22 = 7 \Rightarrow \mathcal{C} = 19$.
- Vérification : $4^{21} \times 21^{19} \bmod 23 = 7^7 \bmod 23 = 5$.

Chapitre 3

Gestion des Clés Publiques (PKI)

L'infrastructure de gestion de clés publiques (PKI : *Public Key Infrastructure*) représente l'ensemble des moyens matériels et logiciels assurant la gestion des clés publiques sur un réseau exposé. Son rôle est de garantir la liaison entre chaque utilisateur à sa clé publique.

3.1 Certificat d'identité

Un certificat d'identité est un document électronique qui garantit un certain nombre d'informations décrivant et identifiant d'une manière sure un utilisateur. Les certificats sont signés et délivrés par une tierce partie de confiance appelée autorité de certification. La structure d'un certificat est normalisée par le standard X.509 :

- ⋆ Version du standard.
- ⋆ Numéro de série du certificat.
- ⋆ L'identité de l'utilisateur.
- ⋆ La clé publique de l'utilisateur.
- ⋆ L'identité du signataire.
- ⋆ Algorithme de signature utilisé.
- ⋆ Fonction de hachage utilisée.
- ⋆ Période de validité du certificat.
- ⋆ Signature numérique de l'autorité de certification.

3.2 Certificat d'attribut

Un certificat d'identité est un document électronique qui donne des informations sur les privilèges d'un utilisateur. Ces informations sont liées aux droits que possède

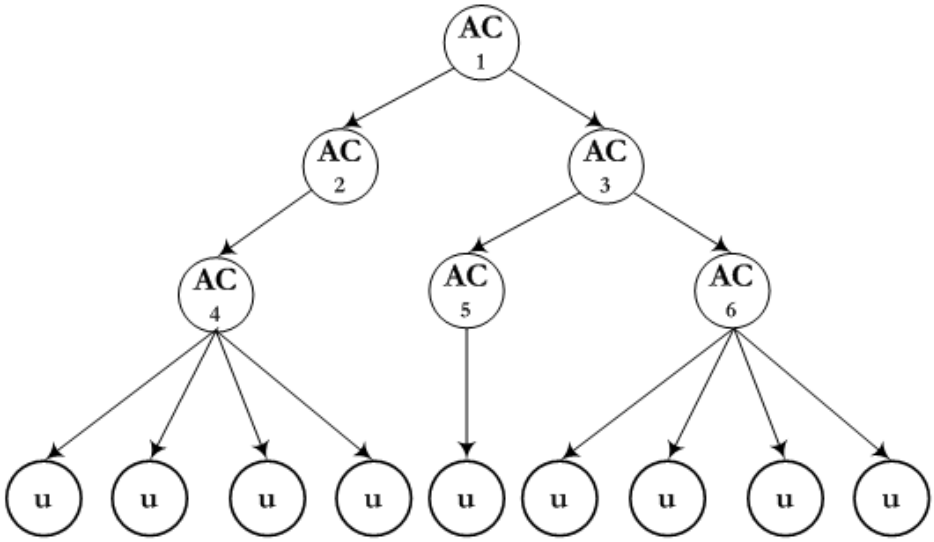
un utilisateur dans une application donnée, qui lui permet de procéder à certaines opérations techniques ou accéder à certains services ou ressources.

3.3 Autorité de certification

L'autorité de certification est un serveur central qui délivre les certificats pour les utilisateurs du système. Elle signe les certificats en utilisant sa propre clé privée pour garantir les informations liées à l'utilisateur. Les certificats générés pour tous les utilisateurs ne peuvent pas être issus d'une même autorité de certification. Ainsi, il est nécessaire que ce rôle soit réparti à travers plusieurs autorités de certification.

3.3.1 Modèle hiérarchique

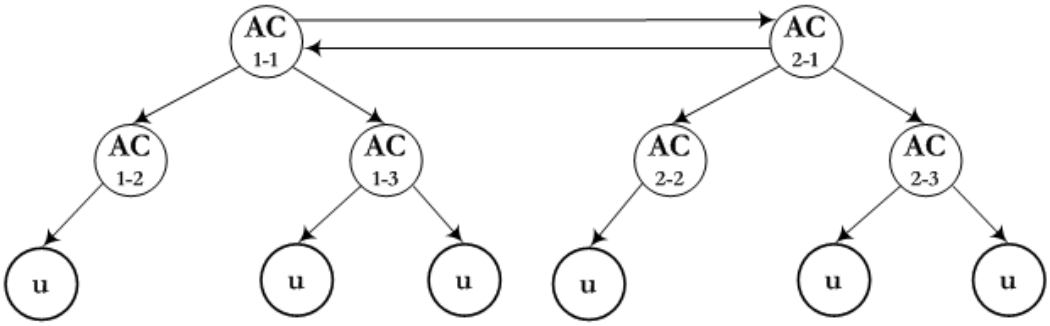
Dans ce modèle, l'autorité de certification racine délivre un certificat d'attribut qui garantit la délégation du service de certification à une ou plusieurs d'autres autorités, qui elles-mêmes à leurs tours délivrent des certificats à d'autres, et ainsi de suite.



3.3.2 Modèle croisé

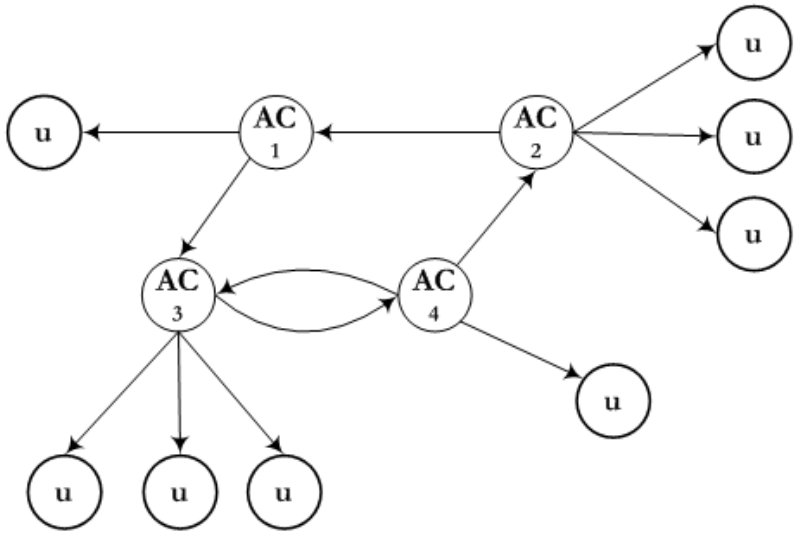
Les relations croisées servent à relier deux hiérarchies d'autorités de certification de deux organisations différentes. L'autorité de certification racine de chaque organisation signe pour l'autre un certificat d'identité. Ainsi, n'importe quel utilisateur de la

première hiérarchie peut vérifier la clé publique de n'importe quel autre utilisateur de la deuxième.



3.3.3 Modèle en graphe

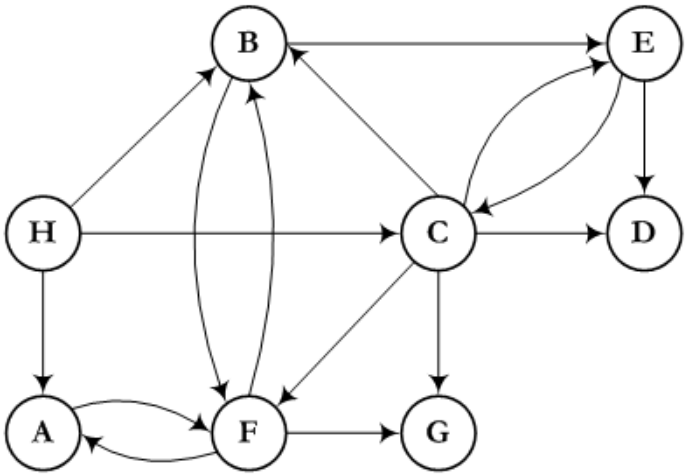
Dans ce modèle, chaque autorité de certification délivre un certificat d'identité à l'autorité en qui elle fait confiance.



3.4 Gestion complètement distribuée des certificats

Dans ce type de système, il n'y a pas la notion d'autorité de certification. Chaque utilisateur se considère comme étant une autorité de certification, où il peut signer et délivrer des certificats pour les utilisateurs. Ce système permet la création d'un graphe de confiance entre l'ensemble des utilisateurs. La vérification de la validité d'un

certificat se fait à travers la vérification de toute la chaîne de certificats qui relie les deux utilisateurs en se basant sur le principe : "Si $\mathcal{A}$ fait confiance à $\mathcal{B}$, et ce dernier fait confiance à $\mathcal{C}$, alors A peut faire confiance à $\mathcal{C}$".



3.5 Révocation des certificats

Un certificat peut être révoqué en cas de :

- ⋆ L'expiration de la période de validité.
- ⋆ La divulgation de la clé privée de l'utilisateur (ce qui sera suivi par la mise à jour de sa paire de clés).
- ⋆ L'utilisateur n'est plus considéré digne de confiance.

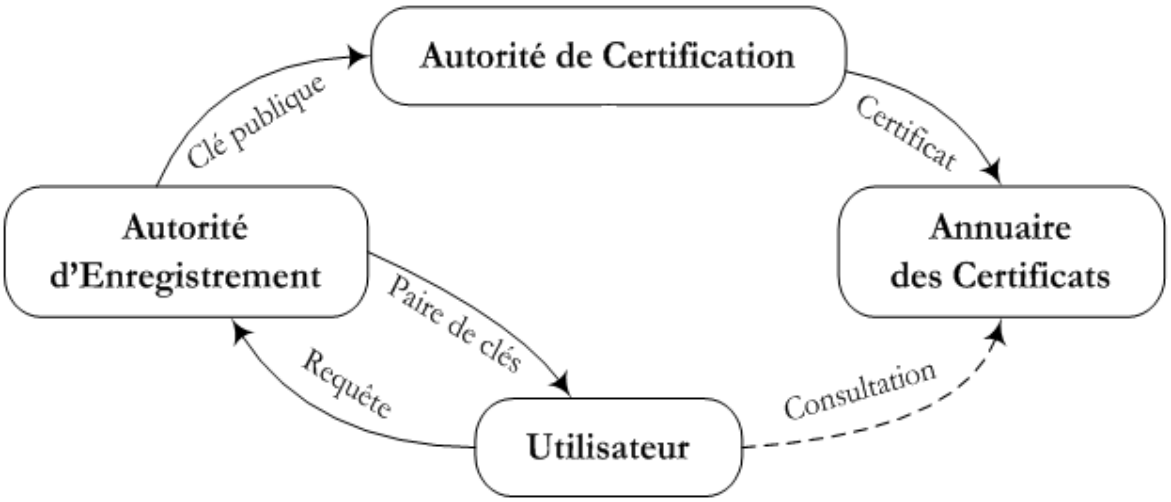
3.6 Les annuaires de certificats

Les certificats émis par les autorités de certification doivent être rendus accessibles afin que les utilisateurs du système puissent y accéder. Pour cela, les certificats sont publiés à travers un annuaire à accès libre. Il constitue en quelque sorte une base de données centrale accessible en mode de lecture, en mode d'ajout (nouveau certificat) et en mode de suppression (révocation d'un certificat). Il contient également une liste de révocation de certificats qui comporte la liste de l'ensemble des certificats révoqués depuis la création de l'annuaire, datés et signés par les autorités de certification.

3.7 La procédure de certification

L'infrastructure de gestion de clés publiques fournit trois services principaux :

- ⋆ La génération et la mise à jour des paires de clés.
- ⋆ La certification des clés publiques et la publication des certificats.
- ⋆ La révocation des certificats.



La procédure de certification se fait principalement en trois étapes :

(1) L'utilisateur s'enregistre auprès de l'autorité d'enregistrement en donnant des justifications concrètes vis-à-vis son identité.

(2) L'autorité d'enregistrement génère la paire de clés de l'utilisateur et envoie une requête de certification à l'autorité de certification.

(3) L'autorité de certification génère un certificat pour l'utilisateur et publie ce dernier sur l'annuaire.

3.8 Le partage du pouvoir de certification

3.8.1 La cryptographie à seuil

Le partage de secret repose sur le concept de détention d'une portion d'une information secrète par plusieurs entités. La cryptographie à seuil permet le partage d'une valeur secrète S à un ensemble de n serveurs, sans que chacun d'eux connaisse sa valeur. A partir d'au moins k serveurs on peut reconstruire le secret ($k \leq n$). Si le

nombre de serveurs est inférieur à k, aucune information n'est obtenue sur le secret S. La cryptographie à seuil s'exécute en deux étapes :

(1) *Le partage du secret :* Cette étape permet de mettre en commun un secret S entre n serveurs. On crée un polynôme $F(x)$ de degré $k-1$ avec des coefficients arbitraires en mettant $a_0 = S$. On choisit ensuite publiquement n points distincts x_i, et on distribue secrètement à chaque serveur une part privée $(x_i, F(x_i))$. Le point x_i pourrait être n'importe quelle valeur publique qui identifie le serveur s_i d'une manière unique. Pour simplifier la notation, on met $x_i = i$, par conséquent les parts privées sont dénotées par $F(1), F(2), ..., F(n)$.

(2) *La reconstruction du secret :* Cette étape permet de reconstruire le secret S à partir d'un sous-ensemble de k parts : $\{F(1), F(2), ..., F(k)\}$. Etant donné k points distincts $(i, F(i))$, il existe un polynôme unique $F(x)$ de degré $k-1$ passant par tous les points. Ce polynôme peut être calculé à partir des points $(i, F(i))$ en utilisant l'interpolation de Lagrange.

3.8.2 Certification avec la cryptographie à seuil à base de RSA

La certification s'exécute comme suit :

(1) Générer les paramètres d'une paire de clés RSA : e, d, n, $\phi(n)$.

(2) Définir un polynôme $\mathcal{F}(x) = d + a_1 x + ... + a_{k-1} x^{k-1}$, ensuite calculer pour chaque serveur s_i sa part privée : $S_i = F(i) \bmod \phi(n)$.

(3) Chaque serveur s_i génère une signature partielle $\mathcal{SP}_i$ du certificat comme suit : $\mathcal{SP}_i = \mathcal{C}^{\mathcal{S}_i} \bmod n$.

(4) Pour combiner l'ensemble des signatures partielles, calculer l'interpolation de Lagrange de chaque serveur : $\mathcal{L}_i = \prod_{j=1, j\neq i}^{j=k} \frac{j}{j-i} \bmod \phi(n)$. Ensuite, calculer la signature complète $\mathcal{SC}$ du certificat : $\mathcal{SC} = \prod_{i=1}^{i=k} \mathcal{SP}_i^{\mathcal{L}_i} \bmod n$.

(5) Si l'égalité $\mathcal{C} = \mathcal{SC}^e \bmod n$ est vérifiée, alors la signature est valide.

3.8.2.1 Exemple

⋆ $e = 11$, $d = 131$, $n = 527$, $\phi(n) = 480$.

⋆ $\mathcal{F}(x) = 131 + x + x^2$. $\mathcal{S}_1 = \mathcal{F}(1) \bmod 480 = 133$, $\mathcal{S}_2 = \mathcal{F}(2) \bmod 480 = 137$, et

$\mathcal{S}_3 = \mathcal{F}(3) \bmod 480 = 143$.

- ⋆ Pour $\mathcal{C} = 6$: $\mathcal{SP}_1 = 6^{133} \bmod 527 = 347$, $\mathcal{SP}_2 = 6^{137} \bmod 527 = 181$, et $\mathcal{SP}_3 = 6^{143} \bmod 527 = 88$.
- ⋆ $\mathcal{L}_1 = 3$, $\mathcal{L}_2 = 477$, et $\mathcal{L}_3 = 1$. $\mathcal{SC} = 347^3 \times 181^{477} \times 88^1 \bmod 527 = 522$.
- ⋆ $522^{11} \bmod 527 = 6$.

Chapitre 4

Gestion des Clés Symétriques (Kerberos)

Kerberos est un système qui permet l'authentification des utilisateurs et des services dans un réseau exposé aux attaques. Il utilise le principe de tierce partie de confiance. Toutes les entités du réseau font confiance à un serveur central de distribution de clés. Kerberos utilise des mécanismes de chiffrement à clés symétriques, et authentifie les utilisateurs à travers le protocole de *Needham et Schroeder*.

4.1 Faille des protocoles

Un protocole est un algorithme distribué faisant intervenir plusieurs participants. Chaque participant exécute ses actions sur sa machine en ayant la possibilité de participer à plusieurs sessions du même protocole (il peut changer de rôle d'une session à une autre). Tous les utilisateurs légitimes qui y participent se comportent conformément à la spécification du protocole à l'exception de l'intrus.

L'intrus peut être un utilisateur régulier qui possède ses propres clés. Il peut écouter les messages qui circulent sur le réseau, supprimer ou modifier ceux qu'il intercepte. Tous les messages envoyés par l'intrus doivent appartenir à sa base de connaissances (à travers ses connaissances initiales et les messages intermédiaires reçus). La topologie du réseau joue un rôle très important qui renforce la capacité de l'intrus. On dit que $\mathcal{I}$ maintient la communication entre $\mathcal{A}$ et $\mathcal{B}$ si $\mathcal{I}$ est l'unique utilisateur qui se trouve physiquement entre les deux. Dans ce cas, $\mathcal{I}$ peut mener une attaque active pour détourner le protocole.

On dit qu'un protocole d'authentification contient une faille si $\mathcal{I}$ arrive à prouver qu'il est autre que $\mathcal{I}$ en exécutant correctement la spécification du protocole. La preuve

de l'existence d'une faille doit être une trace valide de ce protocole montrant que l'objectif visé n'est pas atteint à travers un contre exemple. On distingue deux types de failles :

— *Faille à rôle simple*, là où l'intrus exécute une seule session du protocole.

— *Faille à rôle multiple*, là où l'intrus ouvre plusieurs sessions en parallèle.

4.2 Authentification par secret partagé

La manière la plus basique pour assurer le service d'authentification est d'utiliser les nonces en utilisant le protocole suivant :

1. $\mathcal{A}$ appelle $\mathcal{B}$.
2. $\mathcal{B}$ répond à $\mathcal{A}$ avec un nonce $\mathcal{N}_b$.
3. $\mathcal{A}$ répond à $\mathcal{B}$ avec $\{\mathcal{N}_b\}_{\mathcal{K}_{ab}}$.
4. $\mathcal{B}$ vérifie si $N_b = \{\{\mathcal{N}_b\}_{\mathcal{K}_{ab}}\}_{\mathcal{K}_{ab}}$.

Pour une authentification mutuelle, $\mathcal{A}$ doit émettre à son tour un nonce $\mathcal{N}_a$ que $\mathcal{B}$ doit le retourner chiffré avec la clé $\mathcal{K}_{ab}$. Une autre méthode d'authentification par secret partagé consiste que $\mathcal{B}$ envoie un nonce chiffré avec $\mathcal{K}_{ab}$. $\mathcal{A}$ déchiffre le nonce et effectue sur ce dernier une opération spécifique. Ensuite, il renvoie le résultat chiffré avec la clé $\mathcal{K}_{ab}$. On obtient le protocole suivant :

1. $\mathcal{A}$ appelle $\mathcal{B}$ en envoyant $\{\mathcal{N}_a\}_{\mathcal{K}_{ab}}$
2. $\mathcal{B}$ déchiffre le message pour obtenir $\mathcal{N}_a$, et répond avec $\{\mathcal{N}_b, \mathcal{N}_a + 1\}_{\mathcal{K}_{ab}}$
3. $\mathcal{A}$ déchiffre le message pour obtenir $\mathcal{N}_b$, et répond avec $\{\mathcal{N}_b + 1\}_{\mathcal{K}_{ab}}$

4.3 Protocole de Needham et Schroeder

L'un des inconvénients majeurs du schéma précédent est le nombre de clés préétablies dans le réseau. Pour n utilisateurs et m services, $n \times m$ clés doivent être générées. Needham et Schroeder ont proposé un protocole pour résoudre ce problème en utilisant une tierce partie de confiance (notée TP). Cette dernière partage une clé secrète avec

chaque entité dans le réseau. On note $\mathcal{K}_i$, la clé secrète partagée entre l'entité i et le serveur TP.

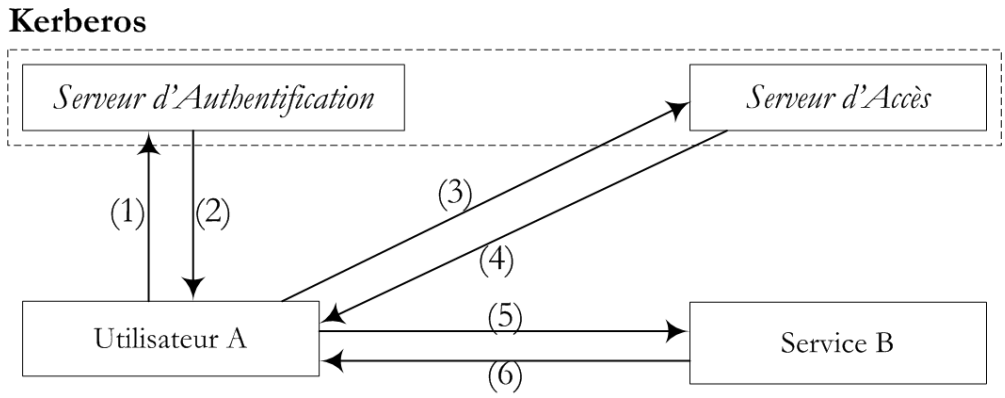
1. $\mathcal{A}$ demande à TP d'accéder à $\mathcal{B}$ et joint $\{\mathcal{N}'_a, B\}_{\mathcal{K}_a}$.
2. TP génère une clé de session $\mathcal{K}_{ab}$ (à partager entre $\mathcal{A}$ et $\mathcal{B}$) et répond $\mathcal{A}$ avec $\{\mathcal{N}'_a + 1, \mathcal{K}_{ab}, \mathcal{B}, \mathcal{T}\}_{\mathcal{K}_a}$, tel que $\mathcal{T} = \{\mathcal{K}_{ab}, \mathcal{A}\}_{\mathcal{K}_b}$.
3. $\mathcal{A}$ déchiffre le message pour avoir $\mathcal{T}$ et $\mathcal{K}_{ab}$, et appelle $\mathcal{B}$ en envoyant $\mathcal{T}$ et $\{\mathcal{N}_a\}_{\mathcal{K}_{ab}}$.
4. $\mathcal{B}$ déchiffre $\mathcal{T}$ avec $\mathcal{K}_b$ pour avoir $\mathcal{K}_{ab}$, ensuite à l'aide de cette dernière déchiffre $\{\mathcal{N}_a\}_{\mathcal{K}_{ab}}$ et répond à $\mathcal{A}$ avec $\{\mathcal{N}_a + 1, \mathcal{N}_b\}_{\mathcal{K}_{ab}}$.
5. $\mathcal{A}$ répond à $\mathcal{B}$ avec $\{\mathcal{N}_b + 1\}_{\mathcal{K}_{ab}}$.

4.4 Fonctionnement de Kerberos

Le système Kerberos composé de deux serveurs : *le serveur d'authentification* et *le serveur d'accès*. Les deux serveurs communiquent à travers un canal sécuritaire et partagent une clé symétrique $\mathcal{K}$. Kerberos partage avec chaque entité i (utilisateur/service) dans le réseau une clé secrète PWi. Un utilisateur demande au serveur d'authentification un ticket TGT (*Ticket Granting Ticket*) pour accéder au serveur d'accès. Ensuite, il demande au serveur d'accès un ticket TGS (*Ticket Granting Service*) pour accéder à un service particulier. Si l'utilisateur possède les droits d'accès à ce service, le serveur d'accès lui délivre le ticket demandé. Ce dernier n'est valable que pour un seul utilisateur et un seul service. Il comporte l'identificateur de l'utilisateur, l'identificateur du serveur, la clé de session, et une date d'expiration ; le tout est chiffré avec la clé secrète du service. L'utilisateur ne peut pas déchiffrer le ticket, mais il l'utilise à chaque fois qu'il désire accéder au service jusqu'à ce que la date de validité soit expirée.

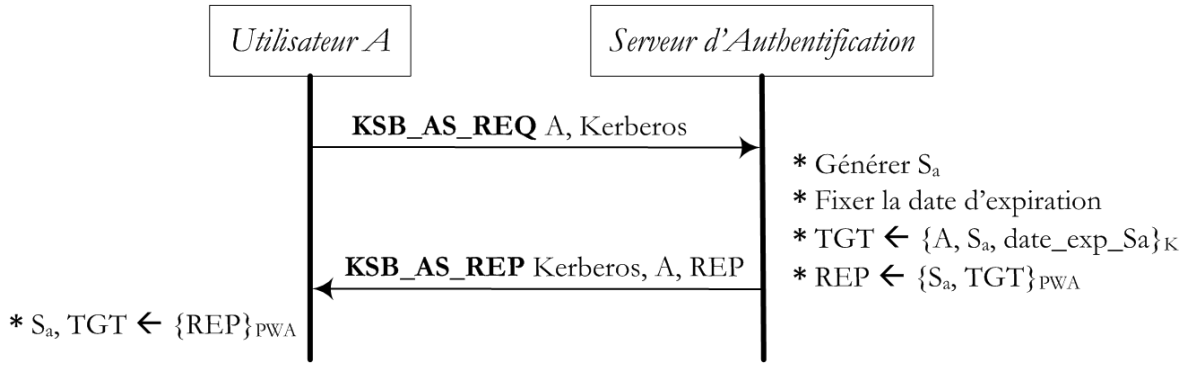
4.4.1 Authentification de l'utilisateur

L'utilisateur $\mathcal{A}$ envoie au serveur d'authentification une requête. Le serveur lui répond avec une clé de session $\mathcal{S}_a$ (qui sera partagée avec le serveur d'accès) et un ticket TGT chiffré avec la clé $\mathcal{K}$ du système Kerberos ; le tout est chiffré avec la clé secrète



(1 & 2) : Authentification de l'utilisateur A. **(3 & 4)** : Autorisation d'accès au service B. **(5 & 6)** : Exécution du service B.

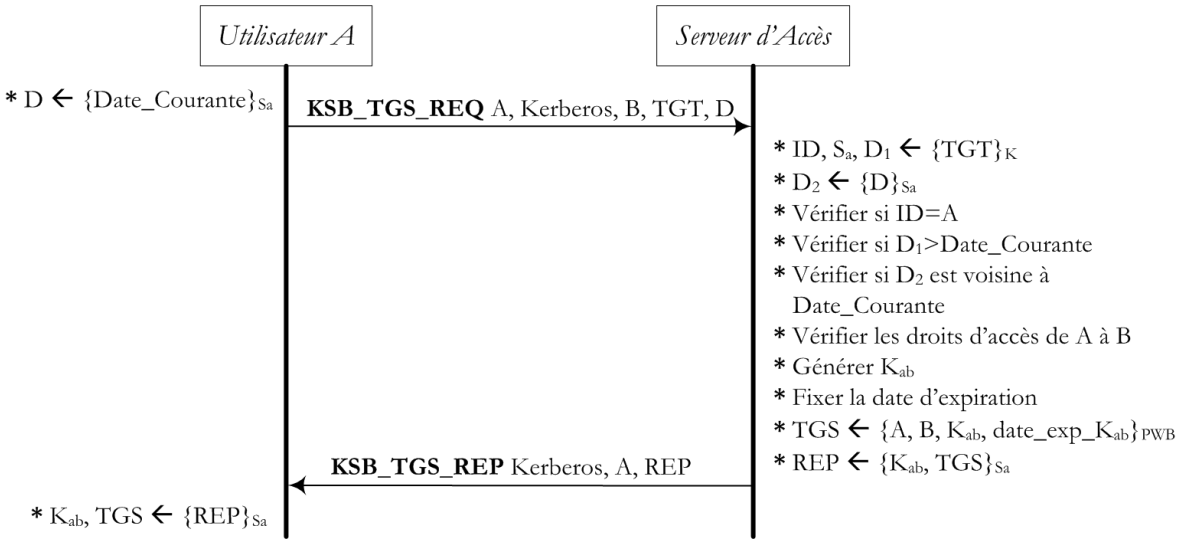
de l'utilisateur $\mathcal{A}$ (PWA). Le ticket TGT comporte l'identificateur de l'utilisateur, la clé de session $\mathcal{S}_a$, et la date d'expiration.



4.4.2 Autorisation d'accès de l'utilisateur

Lorsque l'utilisateur $\mathcal{A}$ désire utiliser le service $\mathcal{B}$, il adresse au serveur d'accès une requête qui comporte l'identificateur de l'utilisateur, l'identificateur du service, le ticket TGT, et la date courante chiffrée avec la clé $\mathcal{S}_a$. Le serveur d'accès déchiffre le ticket TGT en utilisant la clé $\mathcal{K}$, récupère la clé de session $\mathcal{S}_a$, déchiffre la date courante reçue, et vérifie si cette dernière est voisine à sa date courante. Ensuite, il vérifie les droits d'accès de cet utilisateur au service demandé et la date d'expiration du ticket.

Si l'utilisateur possède les droits d'accès, le serveur lui délivre un ticket d'accès TGS qui comporte l'identificateur de l'utilisateur, l'identificateur du service demandé, une clé de session $\mathcal{K}_{ab}$, et une date d'expiration ; le tout chiffré avec la clé secrète du service $\mathcal{B}$ (PWB).



4.4.3 Accès au service

L'utilisateur $\mathcal{A}$ envoie au service $\mathcal{B}$ une requête contenant son identificateur, l'identificateur du service, le ticket d'accès TGS, et la date courante chiffrée avec la clé de session $\mathcal{K}_{ab}$. Le service déchiffre le ticket, récupère la clé $\mathcal{K}_{ab}$, vérifie si le ticket est toujours valide, et vérifie si la date reçue est voisine à celle de sa date courante. Ensuite, le service lance un protocole pour exécuter le service demandé.