Certificats et infrastructures à clés publiques

christina.boura@uvsq.fr

20 avril 2018

1 Attaque de l'homme du milieu

L'*attaque de l'homme du milieu* est une attaque très puissante contre les cryptosystèmes à clé publique. L'idée principale est qu'un attaquant, disons Oscar, remplace les clés publiques envoyées par Alice et Bob par ses propres clés. Oscar peut faire cette attaque contre n'importe quel cryptosystème asymétrique si les clés ne sont pas authentifiées. On étudie ici cette attaque contre le protocole d'échange de clés Diffie-Hellman, mais comme on vient de dire, la même attaque s'applique à n'importe quel système à clé publique.

Le protocole d'échange de clés Diffie-Hellman permet à deux personnes qui ne sont probablement jamais rencontrées par le passé de se mettre d'accord sur une clé partagée afin d'échanger des messages confidentiels à travers un canal publique. On rappelle brièvement les étapes du protocole dans la figure 1.

Alice		**Bob**
Choisir aléatoirement $a = k_{prA}$		
Calculer $A = k_{pubA} \equiv \alpha^a \mod p$		
	$\xrightarrow{A}$	
		Choisir aléatoirement $b = k_{prB}$
		Calculer $B = k_{pubB} \equiv \alpha^b \mod p$
	$\xleftarrow{B}$	
$k_{AB} = B^a \mod p$		$k_{AB} = A^b \mod p$

FIGURE 1 – Le protocole d'échange de clés de Diffie et Hellman

On suppose maintenant qu'Oscar a un rôle actif dans cet échange entre Alice et Bob. Plus précisément, on suppose qu'Oscar est capable d'intercepter les messages échangés (ici les clés publiques d'Alice et de Bob) et les modifier. Ce scénario n'est pas du tout surréaliste et peut facilement se réaliser à travers un canal de communication non-protégé. L'idée principale de l'attaque d'Oscar est qu'il remplace les clés publiques d'Alice et de Bob par sa propre clé publique. Cette attaque est illustrée à la figure 2.

Examinons les clés calculées par Alice, Bob et Oscar. La clé que calcule Alice est

$$k_{AO} = \tilde{B}^a \equiv (\alpha^o)^a \equiv \alpha^{oa} \mod p.$$

Cette clé est identique à la clé que calcule Oscar : $k_{AO} = A^o \equiv (\alpha^a)^o \equiv \alpha^{ao} \mod p$. En même temps Bob calcule

$$k_{BO} = \tilde{A}^b \equiv (\alpha^o)^b \equiv \alpha^{ob} \mod p.$$

Cette clé est la même que la clé d'Oscar $k_{BO} = B^o \equiv (\alpha^b)^o \equiv \alpha^{bo} \mod p$.

On peut remarquer que les deux clés malveillantes qu'Oscar envoie, $\tilde{A}$ et $\tilde{B}$ sont en réalité les mêmes. On leur donne ici des noms différentes pour mettre l'accent sur le fait qu'Alice et Bob pensent avoir reçu les clés de l'un l'autre. Ce qui se passe en réalité est que le protocole Diffie-Hellman est exécuté deux fois : Une fois entre Alice et Oscar et une fois entre Oscar et Bob. Par conséquent, Oscar établie une clé commune avec Alice, la clé k_{AO} et une clé commune avec Bob, k_{BO}. Cependant, **ni Alice ni Bob ne se rendent pas compte qu'ils partagent une clé avec Oscar et pas avec l'un l'autre.**

À partir de maintenant, Oscar a le contrôle sur le trafic chiffré échangé entre Alice et Bob. On peut voir dans la figure 3 comment Oscar peut lire leurs messages chiffrés sans qu'Alice et Bob s'en aperçoivent.

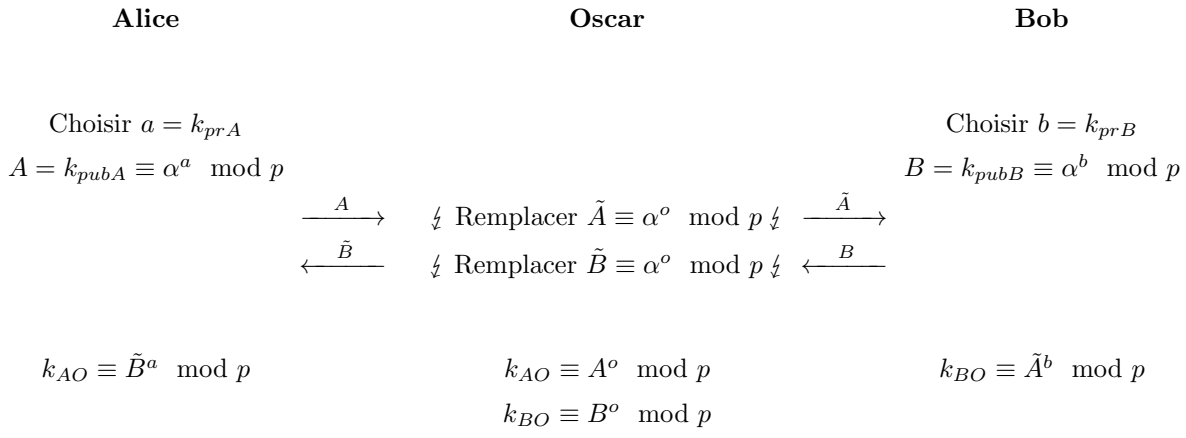

FIGURE 2 – L'attaque de l'homme du milieu contre le protocole Diffie Hellman

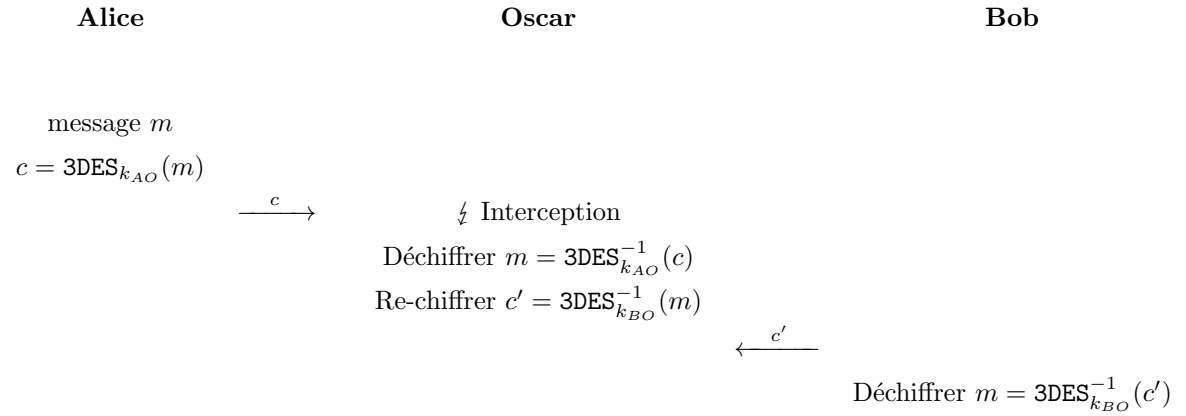

FIGURE 3 – Manipulation des messages après une attaque de l'homme du milieu

Dans le cadre de cet exemple on a supposé que l'algorithme `3DES` a été utilisé pour chiffrer la communication entre Alice et Bob. Évidement, n'importe quel autre algorithme symétrique aurait pu être employé. On note ici qu'Oscar n'est pas seulement capable de lire le contenu d'un message, mais il peut aussi le manipuler. Il peut ensuite chiffrer le nouveau message avec la clé k_{BO} sans que personne s'en doute.

2 Certificats

Le problème sous-jacent de l'attaque de l'homme du milieu est que les clés publiques ne sont pas authentifiées. Dans le cours précèdent nous avons défini la notion de l'authenticité du message qui assure que l'émetteur du message est authentique. Cependant, Bob reçoit une clé publique qu'il suppose être celle d'Alice, mais il n'y a aucun moyen de savoir si ceci est effectivement le cas. Afin d'éclaircir ce point, on peut examiner à quoi la clé d'un utilisateur Alice ressemblerait dans la pratique :

$$k_A = (k_{pubA}, ID_A),$$

où ID_A est un identifiant, par exemple l'adresse IP d'Alice ou le numéro de sa carte étudiant ou encore son nom accompagné de sa date de naissance. La clé publique actuelle k_{pubA} n'est cependant qu'une simple suite binaire (par exemple de 1024 bits dans le cas de RSA-1024). Si Oscar effectue une attaque de l'homme du milieu, il changerait la clé en

$$k_A = (k_{pubO}, ID_A).$$

Puisque tout reste le même à part la suite binaire, le récepteur du message ne va pas pouvoir détecter que la personne de l'autre coté est en réalité Oscar. Afin d'éviter cela, on a besoin d'utiliser des canaux

authentifiés pour distribuer les clés publiques. Le problème de distribution des clés est un problème central de la cryptographie moderne. Il existent plusieurs manières d'adresser ce problème, la plus important étant l'utilisation de *certificats*. L'idée derrière les certificats est assez simple : Puisque l'authenticité du message (k_{pubA}, ID_A) est violée par une attaque active, on applique un mécanisme cryptographique qui assure l'authentification. Plus précisément, on utilise des signatures numériques. Par conséquent, le certificat le plus élémentaire pour un utilisateur Alice a la structure suivante :

$$\text{Cert}_A = [(k_{pubA}, ID_A), \text{sig}_{k_{pr}}(k_{pubA}, ID_A)].$$

L'idée est que le récepteur du certificat vérifie la signature avant d'utiliser la clé publique. Comme on a vu pendant le cours précèdent, la signature protège le message signé (ici la structure (k_{pubA}, ID_A)) contre des manipulations. Si Oscar essaye de remplacer k_{pubA} par k_{pubO} ceci va pouvoir se détecter.

Les certificats nécessitent que le récepteur a la bonne clé de vérification, qui doit être comme on a vu une clé publique. Si on utilisait la clé publique d'Alice, on aurait exactement le même problème que celui qu'on essaie de résoudre. Au lieu de cela, les signatures des certificats doivent être délivrés par un tiers de confiance, mutuellement convenu. Ce tiers de confiance est appelé *Autorité de Certification* (AC ou CA en anglais). C'est la tache de l'autorité de certification de générer et délivrer des certificats pour tous les utilisateurs du système. On peut distinguer deux cas pour la génération des certificats. Dans le premier cas, l'utilisateur calcule sa propre paire de clés et demande à l'autorité de certification de signer sa clé publique, comme ceci est montré dans la figure 4.

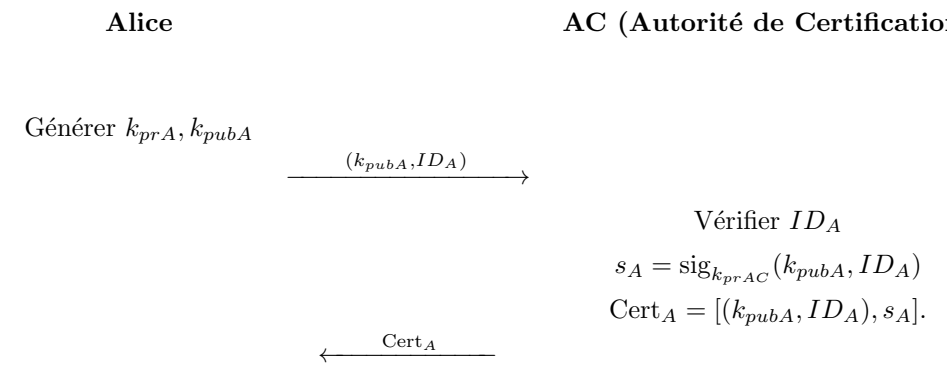

FIGURE 4 – Demande de certification pour des clés générées par l'utilisateur

La première transaction est cruciale pour la sécurité. Il faut s'assurer que le message d'Alice (k_{pubA}, ID_A) est envoyé à travers un canal authentifié. Dans le cas contraire, Oscar peut intercepter le message et demander un certificat au nom d'Alice. Il est souvent plus avantageux que l'autorité de certification signe mais aussi génère les couples clé publique/clé privée pour chaque utilisateur. Un protocole élémentaire pour cette situation est illustré à la figure 5.

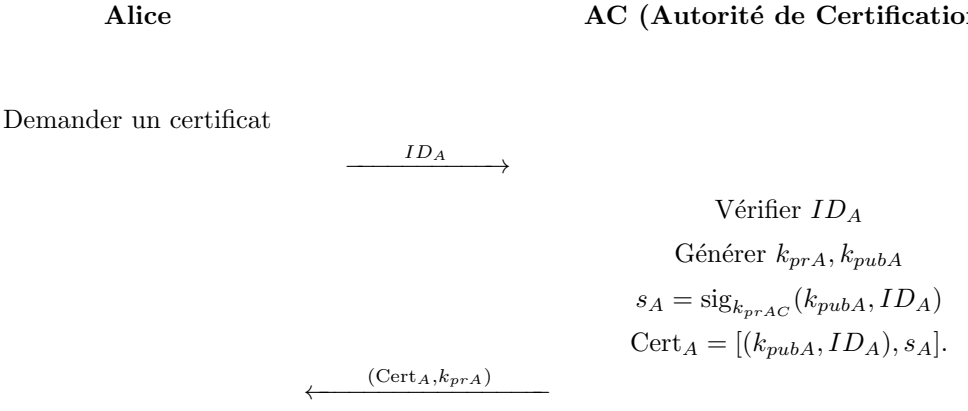

FIGURE 5 – Génération de certificat avec des clés générées par l'autorité de certification

Pour la première transmission, un canal authentifié est nécessaire, puisque l'autorité de certification doit s'assurer que c'est vraiment Alice qui demande le certificat, et pas Oscar qui essaie de se faire passer pour Alice. Le deuxième échange est encore plus important, puisque il fait circuler la clé privée d'Alice en clair. Pour cette raison, un canal sûr doit être utilisé pour cette transmission (par exemple une clé USB envoyé par courrier).

L'échange de clés Diffie-Hellman avec des certificats est décrit dans la figure 6.

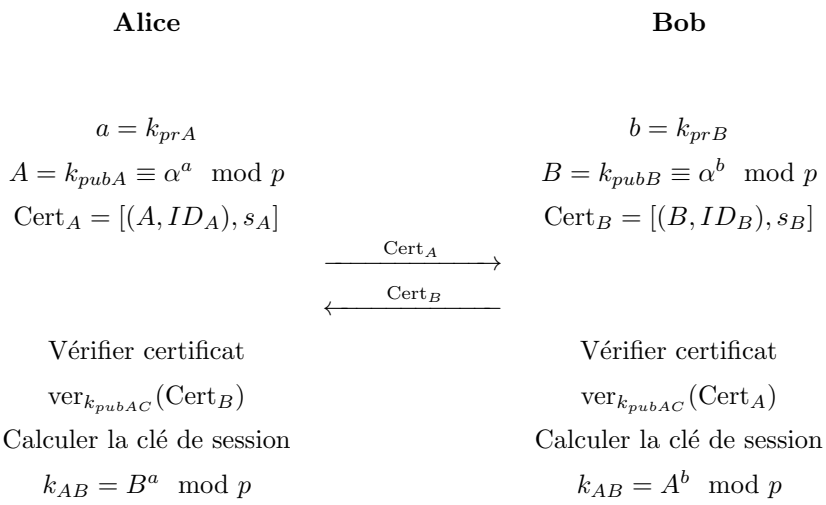

FIGURE 6 – Le protocole d'échange de clés de Diffie et Hellman avec des certificats

Un point très crucial ici est la vérification des certificats. Évidement, sans vérification, les signatures à l'intérieur des certificats seraient sans sens. Comme on peut le voir en suivant le protocole, la vérification nécessite la clé publique de l'autorité de certification. Cette clé doit être transmise à l'intérieur d'un canal authentifié, puisque dans le cas contraire, Oscar pourrait faire une attaque de l'homme du milieu. Même si ceci donne l'impression de n'avoir pas beaucoup avancé depuis le départ, c'est n'est pas comme ça. La différence principale de la situation précédente est que le canal authentifié ne doit être établi qu'une seule fois pendant la phase de mise en place. Aujourd'hui, l'étape de vérification des clés publiques fait souvent partie du logiciel pour ordinateurs, comme les navigateurs web. Dans ce cas le canal authentifié est supposé d'être établi pendant l'installation du logiciel original qui n'a pas été manipulé. Ce qui se passe réellement ici est un *transfert de confiance*. On a vu dans l'exemple du protocole Diffie-Hellman sans certificats, qu'Alice et Bob devaient faire mutuellement confiance à la clé publique de leur correspondant. Avec l'introduction des certificats, ils n'ont besoin de faire confiance qu'à la clé publique k_{pubAC} de l'autorité de certification. Si cette autorité signe également d'autres clés, Alice et Bob savent qu'ils peuvent faire confiance à ces clés sans se soucier. On appelle cela une *chaîne de confiance*.

3 Certificats et infrastructures à clés publiques

L'ensemble formé des autorités de certifications ainsi que des mécanismes nécessaires pour les gérer sont appelés *infrastructures à clé publique* ou PKI (public key infrastracture). Comme on peut l'imaginer, mettre en place et gérer un tel système est une tache très complexe. Des problèmes liés à l'identification des utilisateurs demandeurs des certificats ou à la distribution des clés des autorités de certification doivent être résolus. On peut également énumérer d'autres problèmes liés à l'existence de nombreuses autorités de certification et la révocation des certificats. Certains de ces problèmes sont discutés dans la partie qui suit.

3.1 Les certificats X.509

Dans la pratique, les certificats ne comprennent pas seulement l'identité et la clé publique de l'utilisateur, mais sont composés de plusieurs champs. La norme X.509 est une composante importante des services d'authentification concernant les réseaux et les certificats correspondants sont très utilisés pour les communications à travers internet. Les champs composant un certificat X.509 peuvent être visualisés à la figure 7.

Numéro de série
Algorithme de signature du certificat - Algorithme - Paramètres
Nom du signataire du certificat
Période de validité -Pas avant -Pas après
Détenteur du certificat
Informations sur sa clé publique -Algorithme -Paramètres -Clé publique
Signature

FIGURE 7 – Structure d'un certificat X.509

1. *Algorithme de signature du certificat :* Ce champ spécifie quel algorithme sera utilisé pour la signature, par exemple RSA avec la fonction de hachage SHA-1 (on l'a pas vu pendant ce cours).
2. *Nom du signataire du certificat :* Il existe plusieurs entreprises et organisations émettant des certificats. Ce champ spécifie le nom de l'organisation qui a généré le certificat en question.
3. *Période de validité :* Dans la plupart de cas, une clé publique n'est pas certifiée pour toujours mais seulement pour une courte période, par exemple un ou deux ans. La raison principale pour cela est que la clé privée associée au certificat peut être corrompue. En limitant la période de validité, il n'y a qu'une période de temps très réduite pour qu'un attaquant puisse utiliser de façon malicieuse la clé privée. Une autre raison, qui concerne principalement des certificats pour des entreprises est qu'il peut arriver qu'elles cessent de fonctionner. Si les certificats, et par conséquent les clés publiques ne sont valides que pour une courte durée de vie, alors les dégâts seront limités.
4. *Détenteur du certificat :* Ce champ contient ce que nous avons précédemment appelé ID_A ou ID_B dans nos exemples précédents. Il contient des informations comme des noms des personnes ou des organisations.
5. *Clé publique :* Champ qui contient la clé publique à protéger. À part la suite binaire correspondant à la clé publique, l'algorithme (par exemple Diffie-Hellman) et les paramètres de l'algorithme (par exemple le nombre premier p et l'élément primitif α) y sont stockés.
6. *Signature :* La signature sur tous les champs du certificat.

Il faut ici noter que pour chaque signature deux algorithmes à clé publique sont impliqués : celui dont la clé publique est protégé par le certificat et celui avec qui le certificat est signé. Il peut agir de deux algorithmes complètement différents. Par exemple le certificat peut être signé avec RSA-2048 et la clé publique dans le certificat peut correspondre à un chiffrement Elgamal.

3.2 Chaîne des autorités de certification

Dans un monde idéal, on pourrait imaginer l'existence d'une seule autorité de certification qui délivrerait les certificats pour tous les utilisateurs internet de la planète. Malheureusement, ceci n'est pas le cas. Il existe plusieurs entités agissant comme des autorités de certification. Tout d'abord, plusieurs pays ont leurs propres "officielles" autorités de certification, souvent pour émettre des certificats qui sont utilisés pour des applications liées aux affaires d'état. Ensuite, les certificats pour les sites web sont aujourd'hui délivrés par à peu près 50 autorités commerciales et la plupart des navigateurs ont la clé publique de ces autorités pré-installée. En même temps, plusieurs entreprises délivrent des certificats pour leurs propres employés et des entités externes avec qui elles ont affaire.

Il serait impossible qu'un utilisateur possède les clés de toutes ces différentes autorités de certification. Ce qui est fait en réalité est que les les autorités de certification certifient l'une l'autre.

Regardons un exemple dans lequel le certificat d'Alice est délivré par une autorité AC_1 et celui de Bob par une autorité AC_2. Pour le moment, Alice n'est qu'en possession de la clé publique de son autorité AC_1 et Bob ne connaît que la clé publique de AC_2. Si Bob envoie son certificat à Alice, elle ne vas pas pouvoir vérifier la clé publique de Bob. Cette situation est décrite ci-dessous :

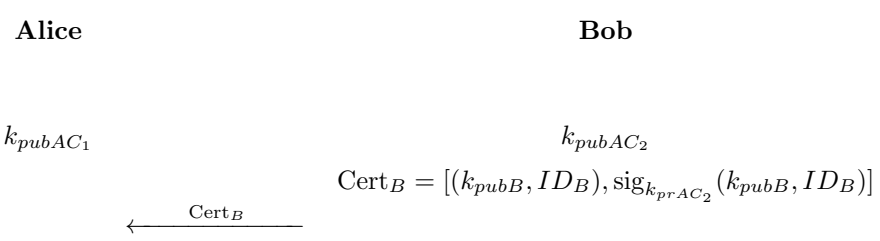

FIGURE 8 – Deux utilisateurs avec des autorités de certification différentes

Alice peut maintenant demander la clé publique de AC_2, qui est elle même contenue dans un certificat signé par l'autorité d'Alice AC_1.

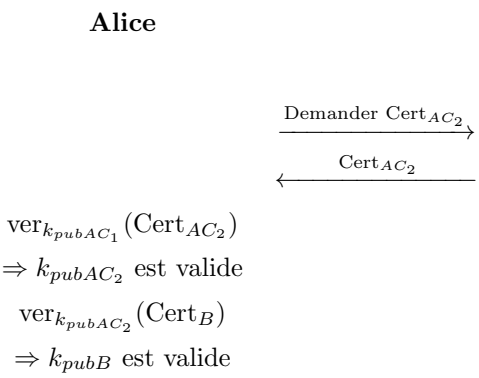

FIGURE 9 – Vérification de la clé publique d'une autorité de certification

La structure Cert_{AC_2} contient la clé privée de AC_2 signée par AC_1, qui ressemble à ça :

$$\text{Cert}_{AC_2} = [(k_{pubAC_2}, ID_{AC_2}), \text{sig}_{k_{prAC_1}}(k_{pubAC_2}, ID_{AC_2})].$$

En suivant cette procédure Alice est maintenant capable de vérifier la clé publique de Bob. On peut alors voir qu'une chaîne de certificats est établie. AC_1 a confiance à AC_2, qui s'exprime par le fait que AC_1 signe la clé publique k_{pubAC_2}. Maintenant Alice peut faire confiance à la clé publique de Bob puisque elle a été signée par AC_1. Cette situation s'appelle *chaîne de confiance* et on dit que *la confiance est déléguée.*

Dans la pratique, les autorités de certification peuvent s'organiser de façon hiérarchique, où chaque autorité signe les clés des autorités du niveau au dessous. Sinon, les autorités peuvent valider les certificats l'une de l'autre de façon *croisée* sans une relation stricte d'hiérarchie.

3.3 Révocation de certificat

Un problème qui doit être réglé dans la pratique est la possibilité de pouvoir révoquer des certificats. Une raison pour cela est que le certificat peut par exemple être entreposé dans une carte à puce qui est perdue. Une autre raison est qu'une personne peut avoir quitté son entreprise et veut s'assurer que sa clé publique n'est pas utilisée par quelqu'un d'autre. La solution à ces situations est facile : Publier une liste de tous les certificats qui ne sont plus valides. Une telle liste est connue sous le nom de *liste de certificats révoqués.* Cette liste contient tous les identifiants des certificats qui ont été révoqués ou ne sont plus valables pour une autorité de certification donnée. Ces certificats ne sont donc plus dignes de confiance. Le problème de ces listes est comment les transmettre aux utilisateurs. La manière la plus simple est

que l'utilisateur contacte l'autorité de certification chaque fois que le certificat d'un nouvel utilisateur est reçu. Le problème de cette approche est que l'autorité de certification doit être impliquée à chaque phase de mise en place. Une façon alternative de gérer cela est que les listes de certificats révoqués sont publiées périodiquement. Le problème est alors qu'il existe toujours une période pendant laquelle un certificat n'est plus valide mais que les utilisateurs ne le sont pas encore au courant. On peut envisager de mettre à jour les listes très souvent, par exemple toutes les heures. Cependant, cette approche risque de ralentir beaucoup le réseau. Un compromis est alors nécessaire et ce qui se fait souvent dans la pratique est que seulement les dernières modifications sont publiées, et non pas les listes entières.