Groupe de travail Réseau
Request for Comments : 2785
Catégorie : Information

R. Zuccherato, Entrust Technologies
mars 2000
Traduction Claude Brière de L'Isle

Méthodes pour éviter les attaques de "petit sous groupe" sur la méthode d'accord de clé Diffie-Hellman pour S/MIME

Statut de ce mémoire
Le présent mémoire apporte des informations pour la communauté de l'Internet. Le présent mémoire ne spécifie aucune sorte de norme de l'Internet. La distribution du présent mémoire n'est soumise à aucune restriction.

Résumé
Dans certaines circonstances, l'utilisation du schéma d'accord de clé Diffie-Hellman dans un sous groupe de premier ordre d'un grand nombre premier p est vulnérable à certaines attaques connues sous le nom d'attaque de "petit sous groupe". Il existe cependant des méthodes pour empêcher ces attaques. Le présent document décrit les situations relevant de la mise en œuvre de S/MIME version 3 dans lesquelles la protection est nécessaire et les méthodes qui peuvent être utilisées pour empêcher ces attaques.

Table des matières

1. Introduction

Le présent document décrit les situations où la protection contre les attaques du type "petit sous groupe" est nécessaire quand on utilise l'accord de clé Diffie-Hellman [RFC2631] dans les mises en œuvre de S/MIME version 3 [RFC2630], [RFC2633]. Donc, les modes éphémère statique et statique-statique de Diffie-Hellman vont être au cœur de notre étude. De possible usages non S/MIME de CMS sont aussi considérés, bien qu'avec moins de détails que les cas relevant de S/MIME. Les situations pour lesquelles la protection est nécessaire sont celles dans lesquelles un attaquant pourrait déterminer une part substantielle (c'est-à-dire plus que quelques bits) de la clé privée d'un utilisateur.

Se protéger contre ces attaques implique certains coûts. Ces coûts peuvent inclure du temps de traitement supplémentaire soit quand une clé publique est certifiée soit quand une clé secrète partagée est déduite, un temps accru pour la génération de paramètres, et éventuellement le recours à des technologies particulières. Tous ces facteurs doivent être pris en compte pour décider de se protéger ou non contre ces attaques, ou de construire l'application de telle sorte que la protection ne soit pas nécessaire.

On ne va pas examiner les "attaques" dans lesquelles l'autre partie à l'accord de clé force simplement la valeur du secret partagé à être "faible" (c'est-à-dire à partir d'un petit ensemble de valeurs possibles) sans tenter de compromettre la clé privée. Il ne vaut pas la peine de tenter d'empêcher ces attaques car l'autre partie dans l'accord de clé obtient le secret partagé et peut simplement rendre public le texte source.

Les méthodes décrites dans le présent mémoire peuvent aussi être utilisées pour fournir une protection contre des attaques similaires sur Diffie-Hellman fondé sur la courbe elliptique.

1.1 Notation

Dans ce document on utilise la même notation que dans la [RFC2631]. En particulier le secret partagé ZZ est généré comme suit :

ZZ = g ^ (xb * xa) mod p

Noter que les parties individuelles effectuent en fait les calculs suivants :

ZZ = (yb ^ xa) mod p = (ya ^ xb) mod p

où ^ note l'exponentiation.

ya est la clé publique de la partie A ; ya = g ^ xa mod p
yb est la clé publique de la partie B ; yb = g ^ xb mod p
xa est la clé privée de la partie A ; xa est dans l'intervalle [2, (q - 2)]
xb est la clé privée de la partie B ; xb est dans l'intervalle [2, (q - 2)]
p est un grand nombre premier
g = h^((p-1)/q) mod p, où h est tout entier avec 1 < h < p-1 tel que h^((p-1)/q) mod p > 1 (g est d'ordre q mod p)
q est un grand nombre premier
j est un grand entier tel que p = q*j + 1

Dans cette discussion, une clé publique "statique" est celle qui est certifiée et est utilisée pour plus d'un accord de clé, et une clé publique "éphémère" est celle qui n'est pas certifiée mais est utilisée une seule fois.

L'ordre d'un entier y modulo p est la plus petite valeur de x supérieure à 1 telle que y^x mod p = 1.

1.2 Brève description de l'attaque

Pour une description complète de ces attaques, voir [LAW] et [LIM].

Si l'autre partie dans une exécution de la méthode d'accord de clé Diffie-Hellman a une clé publique qui n'est pas de la forme décrite ci-dessus, mais de plus petit ordre (où petit signifie moins que q) elle peut alors être capable d'obtenir des informations sur la clé privée de l'utilisateur. En particulier, si des informations sont disponibles sur si un certain déchiffrement a réussi ou non, si le texte chiffré avec la clé objet de l'accord est disponible, ou si un MAC calculé avec la clé objet de l'accord est disponible, des informations sur la clé privée de l'utilisateur peuvent être obtenues.

Supposons que la partie A a une clé publique valide ya et que la partie B a une clé publique yb qui n'est pas de la forme décrite au paragraphe 1.1, mais plutôt que yb est d'ordre r, où r est beaucoup moins que q. Donc yb^r = 1 mod p. Maintenant, quand la partie A produit ZZ comme yb^xa mod p, il va y avoir seulement r valeurs possibles pour ZZ au lieu de q-3 valeurs possibles. À ce point, la partie B ne connaît pas la valeur ZZ, mais peut être capable de faire une recherche exhaustive sur elle.

Si la partie A chiffre le texte en clair avec cette valeur et rend ce texte chiffré disponible à la partie B, la partie B a seulement besoin de faire une recherche exhaustive sur les r possibilités pour déterminer quelle clé a produit le texte chiffré. Quand celui qui est correct est trouvé, cela donne des informations sur la valeur de xa modulo r. De même, si la partie A utilise ZZ pour déchiffrer un texte chiffré et si la partie B est capable de déterminer si le chiffrement a ou non été effectué correctement, les informations sur xa peuvent alors être obtenues. Le nombre réel de messages qui doivent être envoyés ou reçus pour que ces attaques réussissent va dépendre de la structure du nombre premier p. Cependant, il n'est pas déraisonnable de s'attendre à ce que la clé privée entière puisse être déterminée après une centaine de messages.

Une attaque similaire peut être montée si la partie B choisit une clé publique de forme yb = g^xb*f, où f est an élément de petit ordre. Dans cette situation, la partie A va calculer ZZ = yb^xa=g^(xa*xb)*f^xa mod p. Là encore, la partie B peut calculer g^(xa*xb) et peut donc couvrir le petit nombre de valeurs possibles de f^xa mod p pour déterminer les informations sur xa.

Une attaque est aussi possible si la partie B a une clé publique yb d'ordre r où r se met en facteurs dans de petits entiers mais n'est pas nécessairement un petit entier lui-même. Dans ce cas, l'attaquant a besoin de savoir la valeur ZZ calculée par la partie A. À partir de cette valeur, la partie B peut résoudre la clé privée de la partie A modulo r en utilisant l'algorithme Pohlig-Hellman [PH].

Cependant, cette attaque n'est pas aussi praticable que dans les cas présentés précédemment, où les informations sur la clé privée sont récupérées de l'utilisation de ZZ, plutôt que de ZZ lui-même, par une recherche exhaustive.

2. Situations où la protection est nécessaire

Cette Section décrit les situations dans lesquelles l'envoyeur d'un message devrait obtenir une protection contre ce type d'attaque et aussi les situations dans lesquelles le receveur d'un message devrait obtenir la protection. Chaque entité peut décider indépendamment si elle a besoin de protection contre ces attaques.

Cette discussion suppose que la paire de clés du receveur est statique, comme c'est toujours le cas dans la [RFC2631].

2.1 Envoyeur de message

Ce paragraphe décrit les situations dans lesquelles l'envoyeur du message devrait être protégé.

Si la clé de l'envoyeur est éphémère (c'est-à-dire le Diffie- Hellman éphémère-statique est utilisé) aucune protection n'est nécessaire. Dans cette situation, seuls les receveurs du message peuvent obtenir le texte source et le texte chiffré correspondant et donc déterminer les informations sur la clé privée en utilisant les attaques de "petit sous groupe". Cependant, les receveurs peuvent toujours déchiffrer le message et comme la clé de l'envoyeur est éphémère, même si le receveur peut apprendre la clé privée entière, aucun autre message ne court de risque. On remarque ici que si deux receveurs ou plus ont choisi les mêmes paramètres de domaine (p,q,g) alors la même clé publique éphémère peut être utilisée pour tous. Comme la clé est éphémère et seulement associée à un message que les receveurs peuvent déjà déchiffrer, aucune attaque intéressante n'est possible.

Si la clé de l'envoyeur est statique (c'est-à-dire si le Diffie-Hellman statique-statique est utilisé) alors la protection est nécessaire parce que dans cette situation un receveur qui monte une attaque de petit sous groupe peut être capable d'obtenir le texte source d'un autre receveur (peut-être avec une clé publique valide aussi contrôlée par le receveur) et donc pourrait obtenir des informations sur la clé privée. De plus, l'attaquant n'a pas besoin de connaître le texte source pour vérifier si une clé est correcte, pourvu que le texte source ait une redondance suffisante (par exemple, ASCII). Ces informations pourraient alors être utilisées pour attaquer d'autres messages protégés avec la même clé statique.

2.2 Receveur de message

Ce paragraphe décrit des situations dans lesquelles le receveur du message devrait être protégé.

Si absolument aucune information sur le déchiffrement du texte chiffré n'est disponible à une autre partie que le receveur, la protection n'est alors pas nécessaire parce que cette attaque exige des informations sur si le déchiffrement a réussi pour être envoyé à l'attaquant. Donc, aucune mesure de protection n'est nécessaire si la mise en œuvre s'assure qu'aucune information sur le déchiffrement ne peut fuiter. Cependant, la protection peut être une garantie si des utilisateurs humains peuvent donner cette information à l'envoyeur via des moyens hors bande (par exemple des conversations téléphoniques).

Si l'information sur le déchiffrement est disponible à toute autre partie, la protection est alors nécessaire. En particulier, la protection est nécessaire si un événement de protocole permet à toute autre partie de conclure que le déchiffrement a réussi. De tels événements incluent des réponses et le retour de récépissés signés.

3. Méthodes de protection

Cette section décrit cinq mesures de protection que les envoyeurs et les receveurs de messages peuvent utiliser pour se protéger contre les attaques de "petit sous groupe".

Les développeurs devraient noter que certaines des procédures décrites dans cette section peuvent être l'objet de brevets ou de demandes de brevet.

3.1 Validation de clé publique

Cette méthode est décrite au paragraphe 2.1.5 de la [RFC2631], et sa description est répétée ici. Si cette méthode est utilisée, elle devrait être utilisée pour valider les clés publiques de l'autre partie avant de calculer le secret partagé ZZ. La clé publique à valider est y.

1. Vérifier que y se tient dans l'intervalle [2, p-1]. Sinon, la clé est invalide.
2. Calculer y^q mod p. Si le résultat == 1, la clé est valide. Autrement, la clé est invalide.

3.2 CA effectuant la validation de clé publique

L'autorité de certification (CA, *Certification Authority*) pourrait effectuer la méthode de validation de clé publique décrite au paragraphe 3.1 avant de signer et produire un certificat contenant une clé publique Diffie-Hellman. De cette façon, toute partie qui utilise la clé publique peut être assurée qu'un tiers de confiance a déjà effectué le processus de validation de clé. Cette méthode n'est viable que pour les clés publiques statiques. Quand Diffie-Hellman statique-statique est employé, l'envoyeur et le receveur sont tous deux protégés quand la CA a effectué la validation de clé publique. Cependant, quand Diffie-Hellman éphémère-statique est employé, seul l'envoyeur peut être protégé par la validation de clé publique de la CA. Comme l'envoyeur génère une clé publique éphémère, la CA ne peut pas effectuer la validation sur cette clé publique.

Dans le cas d'une clé publique statique, une méthode doit exister pour assurer à l'utilisateur que la CA a bien effectué cette vérification. La CA peut notifier aux utilisateurs de certificat qu'elle a effectué la validation par référence à la politique de certificat (CP, *Certificate Policy*) et à la déclaration de pratique de certificat (CPS, *Certification Practice Statement*) [RFC2527] de la CA ou par des extensions dans le certificat.

3.3 Choix du nombre premier p

Le nombre premier p pourrait être choisi de façon telle que p-1 = 2*q*k où k est un grand nombre premier ou est le produit de grands nombres premiers (grand signifie supérieur ou égal à q). Cela va empêcher un attaquant d'être capable de trouver un élément (autre que 1 et p-1) de petit ordre modulo p, donc cela contrarie l'attaque de petit sous groupe. Une méthode pour produire des nombres premiers de cette forme est de lancer plusieurs fois l'algorithme de génération de nombres premiers jusqu'à ce qu'un nombre premier approprié soit obtenu. Par exemple, la valeur de k pourrait être testée quand à sa primarité. Si k est premier, alors la valeur de p pourrait être acceptée, autrement, l'algorithme de génération de nombres premiers va être relancé, jusqu'à ce qu'une valeur de p soit produite avec k premier.

Cependant, comme avec des nombres premiers de cette forme il y a toujours un élément d'ordre 2 (c'est-à-dire p-1) un bit de la clé privée pourrait encore être perdu. Donc, cette méthode peut n'être pas appropriée dans des circonstances où la perte d'un seul bit de la clé privée pose problème.

Une autre méthode pour produire des nombres premiers de cette forme est de choisir le nombre premier p tel que p = 2*q*k + 1 où k est petit (c'est-à-dire seulement quelques bits). Dans ce cas, la fuite due à une attaque de petit sous groupe va être seulement de quelques bits. Là encore, ceci ne va pas être approprié dans des circonstances où la perte de même quelques bits de la clé privée pose problème. Dans cette approche, q est grand. Noter que dans DSA, q est limité à 160 bits pour des raisons de performances, mais cela n'a pas besoin d'être le cas pour Diffie-Hellman.

De plus, d'autres méthodes (c'est-à-dire de validation de clé publique) peuvent être combinées avec cette méthode afin d'empêcher la perte de quelques bits de la clé privée.

3.4 Exponentiation de cofacteur compatible

Cette méthode de protection est spécifiée dans [P1363] et [KALISKI]. Elle implique de modifier le calcul de ZZ en

incluant j (le cofacteur) dans les calculs et est compatible avec le Diffie-Hellman ordinaire quand les clés publiques des deux parties sont valides. Si la clé publique d'une des parties est invalide, alors le ZZ résultant va être soit 1, soit un élément d'ordre q ; les éléments du petit sous groupe vont être soit détectés, soit annulés. Cette méthode exige que gcd(j,q) = 1.

Au lieu de calculer ZZ comme ZZ = yb^xa mod p, la partie A va le calculer comme ZZ = (yb^j)^c mod p où c = j^(-1)*xa mod q. (La même chose pour la partie B.)

Si la valeur résultante de ZZ satisfait ZZ == 1, alors l'accord de clé devrait être abandonné parce que la clé publique à utiliser est invalide.

Noter que quand j est supérieur à q, comme c'est généralement le cas avec Diffie-Hellman, cette méthode est moins efficace que la méthode du paragraphe 3.1.

3.5 Exponentiation de cofacteur non compatible

Cette méthode de protection est spécifiée dans [P1363]. Comme la méthode du paragraphe 3.4, elle implique de modifier le calcul de ZZ en incluant j (le cofacteur) dans les calculs. Si la clé publique d'une partie est invalide, le ZZ résultant va alors être soit 1, soit un élément d'ordre q ; les éléments de petit sous groupe vont être détectés ou annulés. Cette méthode exige que gcd(j,q) = 1.

Au lieu de calculer ZZ comme ZZ = yb^xa mod p, la partie A va le calculer comme ZZ = (yb^j)^xa mod p. (De même pour la partie B.) Cependant, avec cette méthode, la valeur résultante de ZZ est différente de ce qui est calculé dans la [RFC2631] et donc n'est pas interopérable avec les mises en œuvre conformes à la [RFC2631].

Si la valeur résultante de ZZ satisfait à ZZ==1, l'accord de clé devrait alors être abandonné parce que la clé publique utilisée est invalide.

Note que quand j est supérieur à q, comme c'est généralement le cas avec Diffie-Hellman, cette méthode est moins efficace que la méthode du paragraphe 3.1.

4. Accord de clé éphémère-éphémère

Cette situation est quand l'envoyeur et le receveur d'un message utilisent tous deux des clés éphémères. Bien que cette situation ne soit pas possible dans S/MIME, cela pourrait être utilisé dans d'autres environnements de protocoles. Donc on va brièvement discuter aussi de la protection pour ce cas.

Les développeurs devraient noter que certaines des procédures décrites dans cette section peuvent être l'objet de brevets ou de demandes de brevets.

L'accord de clé éphémère-éphémère donne à un attaquant plus de souplesse car les clés publiques des deux parties peuvent être changées et elles peuvent être contraintes à calculer la même clé à partir d'un petit espace. Cependant, dans le cas éphémère-statique, seule la clé publique de l'envoyeur peut être changée, et seul le receveur peut être contraint par un attaquant extérieur de calculer une clé à partir d'un petit espace.

Donc, dans certains accords de clé éphémère-éphémère, la protection peut être nécessaire pour les deux entités. Une possibilité est que l'attaquant puisse modifier la clé publique des deux parties afin de rendre leur clé partagée prévisible. Par exemple, l'attaquant pourrait remplacer ya et yb par un élément de petit ordre, disons -1. Ensuite, avec une certaine probabilité, l'envoyeur et le receveur vont calculer la même valeur partagée qui vient d'un petit ensemble, facilement couvert.

Noter que dans cette situation, si la protection a été obtenue des méthodes du paragraphe 3.3, alors chaque utilisateur doit s'assurer que la clé publique de l'autre partie ne vient pas d'un petit ensemble d'éléments de petit ordre. Cela peut être fait soit en vérifiant une liste de tels éléments, soit en appliquant de plus les méthodes des paragraphes 3.1, 3.4 ou 3.5.

La protection contre ces attaques n'est cependant pas nécessaire si la clé publique éphémère de l'autre partie a été authentifiée. L'authentification peut être sous la forme d'une signature, d'un MAC, ou de tout autre mécanisme de protection de l'intégrité. Un exemple est celui du protocole de station à station (STS, *Station-To-Station*) [STS]. Comme le

propriétaire authentifie la clé publique, un tiers ne peut pas la modifier et donc ne peut pas monter une attaque. Donc, la seule personne qui pourrait attaquer la clé privée d'une entité est l'autre entité authentifiée dans l'accord de clé. Cependant, comme les deux clés publiques sont éphémères, elles protègent seulement la session en cours à laquelle l'attaquant a accès de toutes façons.

5. Considérations sur la sécurité

Ce document tout entier traite des considérations de sécurité de la mise en œuvre de l'accord de clé Diffie-Hellman.

6. Propriété intellectuelle

L'IETF ne prend pas position sur la validité et la portée de tout droit de propriété intellectuelle ou autres droits qui pourraient être revendiqués au titre de la mise en œuvre ou l'utilisation de la technologie décrite dans le présent document ou sur la mesure dans laquelle toute licence sur de tels droits pourrait être ou n'être pas disponible ; pas plus qu'elle ne prétend avoir accompli aucun effort pour identifier de tels droits. Les informations sur les procédures de l'ISOC au sujet des droits dans les documents de l'ISOC figurent dans les BCP 78 et BCP 79.

Des copies des dépôts d'IPR faites au secrétariat de l'IETF et toutes assurances de disponibilité de licences, ou le résultat de tentatives faites pour obtenir une licence ou permission générale d'utilisation de tels droits de propriété par ceux qui mettent en œuvre ou utilisent la présente spécification peuvent être obtenues sur le répertoire en ligne des IPR de l'IETF à http://www.ietf.org/ipr.

L'IETF invite toute partie intéressée à porter son attention sur tous copyrights, licences ou applications de licence, ou autres droits de propriété qui pourraient couvrir les technologies qui peuvent être nécessaires pour mettre en œuvre la présente norme. Prière d'adresser les informations à l'IETF à ietf-ipr@ietf.org.

7. Références

[KALISKI] B.S. Kaliski, Jr., "Compatible cofactor multiplication for Diffie-Hellman primitives", Electronics Letters, vol. 34, no. 25, 10 décembre 1998, pp. 2396-2397.

[LAW] L. Law, A. Menezes, M. Qu, J. Solinas and S. Vanstone, "An efficient protocol for authenticated key agreement", Technical report CORR 98-05, University of Waterloo, 1998.

[LIM] C.H. Lim et P.J. Lee, "A key recovery attack on discrete log-based schemes using a prime order subgroup", B.S. Kaliski, Jr., editor, Advances in Cryptology - Crypto '97, Lecture Notes in Computer Science, vol. 1295, 1997, Springer-Verlag, pp. 249-263.

[P1363] IEEE P1363, "Standard Specifications for Public Key Cryptography", 1998, travail en cours.

[PH] S.C Pohlig et M.E. Hellman, "An improved algorithm for computing logarithms over GF(p) et its cryptographic significance", IEEE Transactions on Information Theory, vol. 24, 1972, pp. 106-110.

[RFC2527] S. Chokhani, W. Ford, "Cadre pour la politique de certificats d'infrastructure de clés publiques X.509 sur Internet et pour les pratiques de certification", mars 1999. (*Obsolète, voir* RFC3647) *(Information)*

[RFC2630] R. Housley, "Syntaxe de message cryptographique", juin 1999. *(Obsolète, voir* 3369, 3370) *(P.S.)*

[RFC2631] E. Rescorla, "Méthode d'accord de clé Diffie-Hellman", juin 1999. *(P.S.)*

[RFC2633] B. Rmasdell, "Spécification de message S/MIME version 3", juin 1999. (*Obsolète, voir* RFC3851) *(P.S.)*

[STS] W. Diffie, P.C. van Oorschot and M. Wiener, "Authentication and authenticated key exchanges", Designs, Codes and Cryptography, vol. 2, 1992, pp. 107-125.

8. Adresse de l'auteur

Robert Zuccherato
Entrust Technologies
750 Heron Road
Ottawa, Ontario
Canada K1V 1A7

mél : robert.zuccherato@entrust.com

9. Déclaration complète de droits de reproduction

Remerciement
Le financement de la fonction d'édition des RFC est actuellement fourni par l'Internet Society.