Internet Draft R. Zuccherato(Entrust Technologies)
S/MIME Working Group November 1999
expires in six months

Methods for Avoiding the "Small-Subgroup" Attacks on the Diffie-Hellman Key Agreement Method for S/MIME

<draft-ietf-smime-small-subgroup-02.txt>

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Abstract

In some circumstances the use of the Diffie-Hellman key agreement scheme in a prime order subgroup of a large prime p is vulnerable to certain attacks known as "small-subgroup" attacks. Methods exist, however, to prevent these attacks. This document will describe the situations relevant to implementations of S/MIME version 3 in which protection is required and the methods that can be used to prevent these attacks.

1. Introduction

This document will describe those situations in which protection from "small-subgroup" type attacks are required when using Diffie-Hellman key agreement [x942] in implementations of S/MIME version 3 [CMS, MSG]. Thus, the ephemeral-static modes of Diffie-Hellman will be focused on. The situations that require protection are those in which an attacker could determine a substantial portion (i.e. more than a few bits) of a user's private key.

Protecting oneself from these attacks involves certain costs. These costs may include additional processing time either when a public key is certified or a shared secret key is derived, increased parameter generation time, and possibly the licensing of encumbered technologies.

All of these factors must be considered when deciding whether or not to protect oneself from these attacks, or whether to engineer the application so that protection is not required.

We will not consider "attacks" where the other party in the key agreement merely forces the shared secret value to be "weak" (i.e. from a small set of possible values) without attempting to compromise the private key. It is not worth the effort to attempt to prevent these attacks since the other party in the key agreement gets the shared secret and can simply make the plaintext public.

The methods described in this draft may also be used to provide protection from similar attacks on elliptic curve based Diffie-Hellman.

1.1 Notation

In this document we will use the same notation as in [x942]. In particular the shared secret ZZ is generated as follows:

```
     ZZ = g ^ (xb * xa) mod p
```

Note that the individual parties actually perform the computations:

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

where ^ denotes exponentiation.

```
     ya is Party A's public key; ya = g ^ xa mod p
     yb is Party B's public key; yb = g ^ xb mod p
     xa is Party A's private key
     xb is Party B's private key
     p is a large prime
     g = h^((p-1)/q) mod p, where
     h is any integer with 1 < h < p-1 such that h^((p-1)/q) mod p > 1
           (g has order q mod p)
     q is a large prime
     j a large integer such that p=q*j + 1
```

In this discussion, a "static" public key is one that is certified and is used for more than one key agreement, and an "ephemeral" public key is one that is not certified but is used only one time.

The order of an integer y modulo p is the smallest value of x greater than 1 such that y^x mod p = 1.

1.2 Brief Description of Attack

For a complete description of these attacks see [LAW98] and [LIM].

If the other party in an execution of the Diffie-Hellman key agreement method has a public key not of the form described above, but of small order (where small means less than q) then he/she may be able to obtain information about the user's private key. In particular, if information on whether or not a given decryption was successful is available, if ciphertext encrypted with the agreed upon key is available, or if a MAC computed with the agreed upon key is available, information about the

user's private key can be obtained.

Assume Party A has a properly formatted public key ya and that Party B has a public key yb that is not of the form described in Section 1.1, rather yb has order r, where r<<q. Thus yb^r=1 mod p. Now, when Party A produces ZZ as yb^xa mod p, there will only be r possible values for ZZ instead of q possible values. At this point Party B does not know the value ZZ, but may be able to exhaustively search for it.

If Party A encrypts plaintext with this value and makes that ciphertext available to Party B, Party B only needs to exhaustively search through r possibilities to determine which key produced the ciphertext. When the correct one is found, this gives information about the value of xa modulo r. Similarly, if Party A uses ZZ to decrypt a ciphertext and Party B is able to determine whether or not decryption was performed correctly, then information about xa can be obtained. The actual number of messages that must be sent or received for these attacks to be successful will depend on the structure of the prime p. However, it is not unreasonable to expect that the entire private key could be determined after as few as one hundred messages.

A similar attack can be mounted if Party B chooses a public key of the form yb=g^xb*f, where f is an element of small order. In this situation Party A will compute ZZ=yb^xa=g^(xa*xb)*f^xa mod p. Again, Party B can compute g^(xa*xb) and can therefore exhaust the small number of possible values of f^xa mod p to determine information about xa.

2. Situations Where Protection Is Required

This section describes the situations in which the sender of a message should obtain protection against this type of attack and also those situations in which the receiver of a message should obtain protection. Each entity may decide independently whether it requires protection from these attacks.

This discussion assumes that the recipient's key pair is static, as is always the case in [x942].

2.1 Message Sender

This section describes situations in which the message sender should be protected.

If the sender's key is ephemeral, (i.e. ephemeral-static Diffie-Hellman is being used), then no protection is required. In this situation only the recipients of the message can obtain the plaintext and corresponding ciphertext and therefore determine information about the private key using the "small-subgroup" attacks. However, the recipients can always decrypt the message and since the sender's key is ephemeral, even if the recipient can learn the entire private key no other messages are at risk. Notice here that if two or more recipients have selected the same domain parameters (p,q,g) then the same ephemeral public key can be used for all of them. Since the key is ephemeral and only associated with a message that the recipients can already decrypt, no interesting attacks are possible.

If the sender's key is static (i.e. static-static Diffie-Hellman is being used), then protection is required because in this situation a recipient mounting a small-subgroup attack will obtain the plaintext and corresponding ciphertext and therefore could obtain information about the private key using the "small-subgroup" attacks. This information could then be used to attack other messages protected with the same static key.

2.2 Message Recipient

This section describes situations in which the message recipient should be protected.

If absolutely no information on the decryption of the ciphertext is available to any other party than the recipient, then protection is not required because this attack requires information on whether the decryption was successful to be sent to the attacker. So, no protective measures are needed if the implementation ensures that no information about the decryption can leak out. However, protection may be warranted if human users may give this information to the sender via out of band means (e.g. through telephone conversations).

If information on the decryption is available to any other party , then protection is required.

3. Methods Of Protection

This section describes five protective measures that senders and recipients of messages can use to protect themselves from "small-subgroup" attacks.

3.1 Public Key Validation

This method is described in Section 2.1.5 of [x942], and its description is repeated here. If this method is used, it should be used to validate public keys of the other party prior to computing the shared secret ZZ. The public key to be validated is y.

1. Verify that y lies within the interval [2,p-1]. If it does not, the key is invalid.
2. Compute y^q mod p. If the result == 1, the key is valid. Otherwise the key is invalid.

Note that this procedure may be subject to pending patents.

3.2 CA Performs Public Key Validation

The Certification Authority (CA) could perform the Public Key Validation method described in Section 3.1 prior to signing and issuing a certificate containing a Diffie-Hellman public key. In this way, any party using the public key can be assured that a trusted third party has already performed the key validation process. This method is only viable for static public keys. When Static-Static Diffie-Hellman is employed, both the sender and recipient are protected when the CA has performed public key validation. However, when Ephemeral-Static Diffie-Hellman is employed, only the sender can be protected. Since the sender uses an ephemeral public key, the CA cannot perform the validation on that public key.

In the case of a static public key a method must exist to assure the user that the CA has actually performed this verification. The CA can notify certificate users that it has performed the validation by reference to the CA's Certificate Policy (CP) and Certification Practice Statement (CPS) [RFC2527] or through extensions in the certificate.

Note that this procedure may be subject to pending patents.

3.3 Choice of Prime p

The prime p could be chosen such that p-1=2*q*j where j is a large prime or is the product of large primes (large means greater than or equal to q). This will prevent an attacker from being able to find an element of small order modulo p, thus thwarting the small-subgroup attack. One method to produce primes of this form is to run the prime generation algorithm multiple times until an appropriate prime is obtained. As an example, the value of j could be tested for primality. If j is prime, then the value of p could be accepted, otherwise the prime generation algorithm would be run again, until a value of p is produced with j prime.

However, since with primes of this form there is still an element of order 2 (i.e. -1), one bit of the private key could still be lost. Thus, this method may not be appropriate in circumstances where the loss of a single bit of the private key is a concern.

Another method to produce primes of this form is to choose the prime p such that p = 2*q*j + 1 where j is small (i.e. only a few bits). In this case, the leakage due to a small subgroup attack will be only a few bits. Again, this would not be appropriate for circumstances where the loss of even a few bits of the private key is a concern.

Additionally, other methods (i.e. public key validation) can be combined with this method in order to prevent the loss of a few bits of the private key.

3.4 Compatible Cofactor Exponentiation

This method of protection is specified in [p1363] and [KALISKI]. It involves modifying the computation of ZZ. Instead of computing ZZ as ZZ=yb^xa mod p, Party A would compute it as ZZ=(yb^j)^c mod p where c=j^(-1)*xa mod q. (Similarly for Party B.)

If the resulting value ZZ satisfies ZZ==1, then the key agreement should be abandoned because the public key being used is invalid.

Note that this procedure may be subject to pending patents.

3.5 Non-compatible Cofactor Exponentiation

This method of protection is specified in [p1363]. Similar to the method of Section 3.4, it involves modifying the computation of ZZ.

Instead of computing ZZ as ZZ=yb^xa mod p, Party A would compute it as ZZ=(yb^j)^xa mod p. (Similarly for Party B.) However, with this method the resulting ZZ value is different from what is computed in [x942] and therefore is not interoperable with implementations conformant to [x942].

If the resulting value ZZ satisfies ZZ==1, then the key agreement should be abandoned because the public key being used is invalid.

Note that this procedure may be subject to pending patents.

4. Ephemeral-Ephemeral Key Agreement

This situation is when both the sender and recipient of a message are using ephemeral keys. While this situation is not possible in S/MIME, it might be used in other protocol environments. Thus we will briefly discuss protection for this case as well.

In most ephemeral-ephemeral key agreements protection is required for both entities. In this situation a third party attacker could modify the other entity's public key in order to determine the user's private key (as described in Section 1.2). Another possibility is that the attacker could modify both parties' public key so as to make their shared key predictable. For example, the attacker could replace both ya and yb with some element of small order, say -1. Then, with a certain probability, both the sender and receiver would compute the same shared value that comes from some small, easily exhaustible set.

Note that in this situation if protection was obtained from the methods of Section 3.3, then each user must ensure that the other party's public key does not come from the small set of elements of small order. This can be done either by checking a list of such elements, or by additionally applying the methods of Sections 3.1, 3.4 or 3.5.

Protection from these attacks is not required however if the other party's ephemeral public key has been signed by the other party. An example of this is in the Station-To-Station protocol [STS]. Since the owner authenticates the public key, a third party cannot modify it and therefore cannot mount an attack. Thus, the only person that could attack an entity's private key is the other authenticated entity in the key agreement. However, since both public keys are ephemeral, they only protect the current session that the attacker would have access to anyway.

5. Security Considerations

This entire document addresses security considerations in the implementation of Diffie-Hellman key agreement.

6. Intellectual Property Rights

The IETF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made

any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

7. References

[RFC2527] S. Chokhani and W. Ford, "Internet X.509 Public Key Infrastructure, Certificate Policy and Certification Practices Framework", RFC 2527, March 1999.

[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.

[CMS] R. Housley, "Cryptographic Message Syntax", draft-ietf-smime-cms-XX.txt, work in progress.

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

[LAW98] 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 and 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, work in progress.

[MSG] B. Ramsdell, "S/MIME Version 3 Message Specification", draft-ietf-smime-msg-0X.txt, work in progress.

[x942] E. Rescorla, "Diffie-Hellman Key Agreement Method", draft-ietf-smime-x942-0X.txt, work in progress.

8. Author's Address

Robert Zuccherato

Entrust Technologies
750 **Heron Road**
Ottawa, Ontario
Canada K1V 1A7
robert.zuccherato@entrust.com

Appendix A. Full Copyright Statement