Network Working Group
Request for Comments: 3218
Category: Informational

E. Rescorla
RTFM, Inc.

January 2002

Preventing the Million Message Attack on Cryptographic Message Syntax

Status of this Memo

Copyright Notice

Abstract

This memo describes a strategy for resisting the Million Message Attack.

Table of Contents

1. Introduction 1
2. Overview of PKCS-1 2
2.1. The Million Message Attack 3
2.2. Applicability 3
2.2.1. Note on Block Cipher Padding 4
2.3. Countermeasures 4
2.3.1. Careful Checking 4
2.3.2. Random Filling 5
2.3.3. OAEP 5
2.4. Security Considerations 6
3. Acknowledgments 6
4. References 6
5. Author's Address. 6
6. Full Copyright Statement 7

1. Introduction

When data is encrypted using RSA it must be padded out to the length of the modulus -- typically 512 to 2048 bits. The most popular technique for doing this is described in [PKCS-1-v1.5]. However, in 1998 Bleichenbacher described an adaptive chosen ciphertext attack on SSL [MMA]. This attack, called the Million Message Attack, allowed the recovery of a single PKCS-1 encrypted block, provided that the

attacker could convince the receiver to act as a particular kind of oracle. (An oracle is a program which answers queries based on information unavailable to the requester (in this case the private key)). The MMA is also possible against [CMS]. Mail list agents are the most likely CMS implementations to be targets for the MMA, since mail list agents are automated servers that automatically respond to a large number of messages. This document describes a strategy for resisting such attacks.

2. Overview of PKCS-1

The first stage in RSA encryption is to map the message to be encrypted (in CMS a symmetric content-encryption key (CEK)) into an integer the same length as (but numerically less than) the RSA modulus of the recipient's public key (typically somewhere between 512 and 2048 bits). PKCS-1 describes the most common procedure for this transformation.

We start with an "encryption block" of the same length as the modulus. The rightmost bytes of the block are set to the message to be encrypted. The first two bytes are a zero byte and a "block type" byte. For encryption the block type is 2. The remaining bytes are used as padding. The padding is constructed by generating a series of non-zero random bytes. The last padding byte is zero, which allows the padding to be distinguished from the message.

Once the block has been formatted, the sender must then convert the block into an integer. This is done by treating the block as an integer in big-endian form. Thus, the resulting number is less than the modulus (because the first byte is zero), but within a factor of 2^16 (because the second byte is 2).

In CMS, the message is always a randomly generated symmetric content-encryption key (CEK). Depending on the cipher being used it might be anywhere from 8 to 32 bytes.

There must be at least 8 bytes of non-zero padding. The padding prevents an attacker from verifying guesses about the encrypted message. Imagine that the attacker wishes to determine whether or not two RSA-encrypted keys are the same. Because there are at least 255^8 (about 2^64) different padding values with high probability two encryptions of the same CEK will be different. The padding also prevents the attacker from verifying guessed CEKs by trial-encrypting them with the recipient's RSA key since he must try each potential

pad for every guess. Note that a lower cost attack would be to exhaustively search the CEK space by trial-decrypting the content and examining the plaintext to see if it appears reasonable.

2.1. The Million Message Attack

The purpose of the Million Message Attack (MMA) is to recover a single plaintext (formatted block) given the ciphertext (encrypted block). The attacker first captures the ciphertext in transit and then uses the recipient as an oracle to recover the plaintext by sending transformed versions of the ciphertext and observing the recipient's response.

Call the ciphertext C. The attacker then generates a series of integers S and computes C'=C*(S^e) mod n. Upon decryption, C' produces a corresponding plaintext M'. Most values of M' will appear to be garbage but some values of M' (about one in 2^16) will have the correct first two bytes 00 02 and thus appear to be properly PKCS-1 formatted. The attack proceeds by finding a sequence of values S such that the resulting M' is properly PKCS-1 formatted. This information can be used to discover M. Operationally, this attack usually requires about 2^20 messages and responses. Details can be found in [MMA].

2.2. Applicability

Since the MMA requires so many messages, it must be mounted against a victim who is willing to process a large number of messages. In practice, no human is willing to read this many messages and so the MMA can only be mounted against an automated victim.

The MMA also requires that the attacker be able to distinguish cases where M' was PKCS-1 formatted from cases where it was not. In the case of CMS the attacker will be sending CMS messages with C' replacing the wrapped CEK. Thus, there are five possibilities:

1. M' is improperly formatted.
2. M' is properly formatted but the CEK is prima facie bogus (wrong length, etc.)
3. M' is properly formatted and the CEK appears OK. A signature or MAC is present so integrity checking fails.
4. M' is properly formatted and no integrity check is applied. In this case there is some possibility (approximately 1/32) that the CBC padding block will verify properly. (The actual probability depends highly on the receiving implementation. See "Note on Block Cipher Padding" below). The message will appear OK at the CMS level but will be bogus at the application level.

5. M' is properly formatted and the resulting CEK is correct. This is extremely improbable but not impossible.

The MMA requires the attacker to be able to distinguish case 1 from cases 2-4. (He can always distinguish case 5, of course). This might happen if the victim returned different errors for each case. The attacker might also be able to distinguish these cases based on timing -- decrypting the message and verifying the signature takes some time. If the victim responds uniformly to all four errors then no attack is possible.

2.2.1. Note on Block Cipher Padding

[CMS] specifies a particular kind of block cipher padding in which the final cipher block is padded with bytes containing the length of the padding. For instance, a 5-byte block would be padded with three bytes of value 03, as in:

```
XX XX XX XX XX 03 03 03
```

[CMS] does not specify how this padding is to be removed but merely observes that it is unambiguous. An implementation might simply get the value of the final byte and truncate appropriately or might verify that all the padding bytes are correct. If the receiver simply truncates then the probability that a random block will appear to be properly padded is roughly 1/32. If the receiver checks all the padding bytes, then the probability is 1/256 + (1/256^2) + ... (roughly 1/255).

2.3. Countermeasures

2.3.1. Careful Checking

Even without countermeasures, sufficiently careful checking can go quite a long way to mitigating the success of the MMA. If the receiving implementation also checks the length of the CEK and the parity bits (if available) AND responds identically to all such errors, the chances of a given M' being properly formatted are substantially decreased. This increases the number of probe messages required to recover M. However, this sort of checking only increases the workfactor and does not eliminate the attack entirely because some messages will still be properly formatted up to the point of keylength. However, the combination of all three kinds of checking (padding, length, parity bits) increases the number of messages to the point where the attack is impractical.

2.3.2. Random Filling

The simplest countermeasure is to treat misformatted messages as if they were properly PKCS-1 formatted. When the victim detects an improperly formatted message, instead of returning an error he substitutes a randomly generated message. In CMS, since the message is always a wrapped content-encryption key (CEK) the victim should simply substitute a randomly generated CEK of appropriate length and continue. Eventually this will result in a decryption or signature verification error but this is exactly what would have happened if M' happened to be properly formatted but contained an incorrect CEK. Note that this approach also prevents the attacker from distinguishing various failure cases via timing since all failures return roughly the same timing behavior. (The time required to generate the random-padding is negligible in almost all cases. If an implementation has a very slow PRNG it can generate random padding for every message and simply discard it if the CEK decrypts correctly).

In a layered implementation it's quite possible that the PKCS-1 check occurs at a point in the code where the length of the expected CEK is not known. In that case the implementation must ensure that bad PKCS-1 padding and ok-looking PKCS-1 padding with an incorrect length CEK behave the same. An easy way to do this is to also randomize CEKs that are of the wrong length or otherwise improperly formatted when they are processed at the layer that knows the length.

Note: It is a mistake to use a fixed CEK because the attacker could then produce a CMS message encrypted with that CEK. This message would decrypt properly (i.e. appear to be a completely valid CMS application to the receiver), thus allowing the attacker to determine that the PKCS-1 formatting was incorrect. In fact, the new CEK should be cryptographically random, thus preventing the attacker from guessing the next "random" CEK to be used.

2.3.3. OAEP

Optimal Asymmetric Encryption Padding (OAEP) [OAEP, PKCS-1-v2] is another technique for padding a message into an RSA encryption block. Implementations using OAEP are not susceptible to the MMA. However, OAEP is incompatible with PKCS-1. Implementations of S/MIME and CMS must therefore continue to use PKCS-1 for the foreseeable future if they wish to communicate with current widely deployed implementations. OAEP is being specified for use with AES keys in CMS so this provides an upgrade path to OAEP.

2.4. Security Considerations

This entire document describes how to avoid a certain class of attacks when performing PKCS-1 decryption with RSA.

3. Acknowledgments

Thanks to Burt Kaliski and Russ Housley for their extensive and helpful comments.

4. References

[CMS] Housley, R., "Cryptographic Message Syntax", RFC 2630, June 1999.

[MMA] Bleichenbacher, D., "Chosen Ciphertext Attacks against Protocols based on RSA Encryption Standard PKCS #1", Advances in Cryptology -- CRYPTO 98.

[MMAUPDATE] D. Bleichenbacher, B. Kaliski, and J. Staddon, "Recent Results on PKCS #1: RSA Encryption Standard", RSA Laboratories' Bulletin #7, June 26, 1998.

[OAEP] Bellare, M., Rogaway, P., "Optimal Asymmetric Encryption Padding", Advances in Cryptology -- Eurocrypt 94.

[PKCS-1-v1.5] Kaliski, B., "PKCS #1: RSA Encryption, Version 1.5.", RFC 2313, March 1998.

[PKCS-1-v2] Kaliski, B., "PKCS #1: RSA Encryption, Version 2.0", RFC 2347, October 1998.

5. Author's Address

Eric Rescorla
RTFM, Inc.
2064 Edgewood Drive
Palo Alto, CA 94303

Phone: (650) 320-8549
EMail: ekr@rtfm.com

6. Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.