An Attack on RSA Digital Signature

An attack has been found on some implementations of RSA digital signatures using the padding scheme of PCKS-1 when the public key $e = 3$. Details of the attack on PKCS-1 implementations are provided below. A similar attack could also be applied to implementations of digital signatures as specified in American National Standard (ANS) X9.31. Note that this attack is not on the RSA algorithm itself, but on improper implementations of the signature verification process.

The testing used by the Cryptographic Algorithm Validation Program (CAVP) has been modified to test for this improper implementation on new RSA signature applications that undergo CMVP testing. However, many implementations that have already been certified as implementing RSA signatures using PKCS-1 might have been implemented with this problem.

NIST has designed a sequence of messages that can be used by a vendor to test the vulnerability of an implementation to this type of attack (see http://csrc.nist.gov/cryptval/anncmnts.htm.). Concerned users should contact the vendor of their RSA digital signature application to request information on the vulnerability of their implementation.

Details of the Attack

At the rump session of Crypto 2006, Daniel Bleichenbacher gave an attack on RSA digital signature. The attack works on RSA digital signature with public exponent $e = 3$ and PCKS-1 padding. The following is a brief explanation of the attack.

Background:

A PKCS-1 digital signature is computed on a hash value H(M) that is padded as follows:

00 01 FF FF …FF 00 || ASN.1 || H(M),

where 00 01 FF FF …FF 00 is a padding value, ASN.1 is used to provide information about the hash function (basically, the length of the hash value), and H(M) is the hash value. Note that the hash value H(M) is supposed to be right-justified. In this case, after the digital signature is decrypted using the public exponent $e = 3$, the padded message shown above should be obtained. The hash value H(M) can be extracted by searching past the padding and the ASN.1 values, and selecting the appropriate number of bytes that follow. One way to verify the signature is to compare the extracted value of H(M) with a separately computed hash value on the received message M. If they compare, then the digital signature is considered valid.

X9.31 specifies a similar padding method. The only difference is, instead of being right-justified, the hash value is followed by a 2 byte trailer with a fixed value. When SHA-1 is used, the padded hash value is as follows:

6B BB BB ...BB BA || H(M) || 33 CC

Problem:

Some implementations extract the number of bits for the hash value by their position relative to the padding without checking for unexpected data after the hash value. In the case of PKCS-1, the hash value is selected by finding the end of the padding and the ASN.1 value, and then extracting

the hash value without checking whether or not additional data follows the hash value (i.e., whether or not the hash value is right justified in the padded hash string). In the case of ANS X9.31, the hash value is selected by finding the end of the padding, and then extracting the hash value without checking that only two bytes with the expected values follow the hash value in the padded hash string.
.

Attack:

When the PKCS-1 padding method is used, for any message *M*" with hash value H(*M*"), it is rather easy to find a cubic root of a string like

00 01 FF …FF 00 || ASN.1 || H(*M*") || *garbage*

where the number of occurrences of FF in the padding is reduced, and *garbage* is cleverly chosen to make the modified string into a cube of some value.

When the ANS X9.31 padding method is used, the padded hash could be altered as follows by reducing a number of occurrences of BB in the padding:

6B BB BB ...BB BA || H(*M*) || *garbage*

where the last two bytes of *garbage* are the expected trailer (e.g., 33 CC), and the modified padded hash string has a cubic root.

The attack was presented with $e = 3$ as an example. For both padding methods, whenever a small value of *e* is used, and an e^{th} root of a string like the above can be found, then the attack will apply. When *e* is large, finding an e^{th} root modulo *n* will be hard.

How to prevent the attack when PKCS-1 is used:

- Do not use 3 as the public exponent for RSA Signatures.
- If the PKCS-1 padding is used to find the hash value, then verify that there is no more data on the right of the hash value.

How to prevent the attack when ANS X9.31 is used:

- Do not use 2 or 3 as the public exponent for Digital Signatures.
- If the ANS X9.31 padding is used to locate the hash value, then verify that the hash value is followed by only two bytes containing the expected value of the trailer.