Breaking Public Keys
How to Determine an Unknown RSA Public Modulus

Hans-Joachim Knobloch
Secorvo Security Consulting GmbH

<hans-joachim.knobloch@secorvo.de>

Version 1.0
11. October 2012

Abstract

Not surprisingly, the common use of any public key crypto system involves publishing the public key and keeping the private key secret. There are however a few applications where both the private and public key are kept secret, thereby effectively converting a public key crypto algorithm to a symmetric algorithm.

We show that if the RSA cryptosystem is used in such a symmetric application, it is possible to determine the public RSA modulus if the public exponent is known and short, such as 3 or F_4=65537, and two or more plaintext/ciphertext (or, if RSA is used for signing, signed value/signature) pairs are known.

1 Introduction

Despite the nature of a public key crypto system to publish the public key, there are some rare applications where a public key crypto algorithm is effectively converted to a symmetric algorithm by keeping secret both the private and public key. In the case of the RSA crypto system [1], at least the public modulus is kept secret by these applications, whereas the public exponent is often known or can be guessed.

One example of such an odd RSA application was used during the introduction of DNSSEC [2] in the DNS root zone. In order to prevent premature verification of DNSSEC by clients and to allow for a traceless roll-back by elimination of the DNSSEC records in case any operational problems should occur, the construction of a Deliberately Unvalidatable Root Zone (DURZ) was used. In the DURZ the RSA modulus in the DNSSEC key records was replaced by an appropriate human-readable message. Nevertheless the respective other records were signed with the private key associated with the original modulus.

A second example is the voucher system of the captive portal for the pfSense open source firewall [4], where 64 bit RSA is used as a 64 bit block cipher to encrypt voucher roll and ticket number along with some additional information. The result of this encryption is the voucher code that users have to enter at the captive portal in order to be granted network access. 64 bit block size – and hence the RSA key length – were chosen, because the resulting voucher code must be manually typed by the user and longer voucher codes are not deemed usable. Since 64 bit numbers are susceptible to factoring, the RSA modulus is kept secret by the voucher system.

In both cases the de-facto standard public exponent F_4=65537 is used.

2 The Attack

We will now present a known-plaintext attack that allows to determine the unknown public RSA modulus n given at least two pairs (m_i, c_i) of plaintext message m_i and resulting ciphertext c_i, provided the public exponent e is known (or suspected) and short, such as F_4. By definition of the RSA crypto system [1] they are related through the following congruence:

(1) $c_i \equiv m_i^e \mod n$.

According to definition of modular reduction (1) implies directly that

(2) $m_i^e - c_i = x \cdot n$

for some non-negative integer x.

Note that if $e > \ln_2 n$, i. e. for all reasonable-sized RSA public keys with the commonly used public exponent F_4, x will be a positive integer unless $m_i = 0$ or $m_i = 1$. Even for the smallest possible public exponent 3, only a few hundred more plaintext values m_i will satisfy $m_i^e < n$ and thus be unusable for the following step. So for all practical purposes we can assume that $m_i^e - c_i$ is a proper multiple of n

Note furthermore that for relatively small exponents the value $m_i^e - c_i$ can be computed and handled in the memory of every modern computer.

Now the know-plaintext attack is straight forward: Given two pairs (m_1, c_1) and (m_2, c_2) with the above property (1) (and $m^e > n$) compute their greatest common divisor

(3) $g = \gcd(m_1^e - c_1, m_2^e - c_2)$

For all practical cases, g is either n itself or a small multiple thereof.

If g is not equal to n, either the gcd of g and $m_3^e - c_3$ can be calculated (if more than two plaintext/ciphertext pairs are available) or small cofactors in g can be eliminated by trial division.

Once n is determined, it may be attacked using state-of-the-art factoring algorithms.[1]
In particular relatively small moduli like the 64 bit key used by the pfSense voucher system can be factored, compromising the private key in addition to the secret “public” one.

3 Conclusion

We have shown that it is not generally advisable to transform a public key crypto algorithm to a symmetric one by keeping the public key secret.

In particular users of the pfSense voucher system are urgently recommended to choose the magic number employed in generating and verifying the voucher codes (see [4]) at random, keep it secret and change it regularly in order to prevent the known-plaintext attack described above. Alternatively pfSense could replace RSA by a genuine 64 bit block cipher like Triple DES [5].

---

1 The current public world record is the factorization of a 768 bit RSA modulus [6].

4 References

[1] R.L. Rivest, A. Shamir, and L. Adleman: *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems,1978,* http://people.csail.mit.edu/rivest/Rsapaper.pdf

[2] R. Arends, R. Austein, M. Larson, D. Massey, S. Rose: *RFC 4033 - DNS Security Introduction and Requirements*, March 2005*,* http://www.rfc-editor.org/rfc/rfc4033.txt

[3] J. Abley, D. Knight, M. Larson: *DNSSEC Deployment for the Root Zone (Draft)*, March 2010, http://www.root-dnssec.org/wp-content/uploads/2010/05/draft-icann-dnssec-deployment-02.txt

[4] pfSense project: *Captive Portal Vouchers*, Wiki entry as of 10/10/2012 http://doc.pfsense.org/index.php/Captive_Portal_Vouchers

[5] NIST: *FIPS PUB 46-3 - Data Encryption Standard (DES)* (withdrawn), 1999, http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf

[6] T. Kleinjung, K. Aoki, J. Franke, A. K. Lenstra, E. Thomé, J. W. Bos, P. Gaudry, A. Kruppa, P. L. Montgomery, D. A. Osvik, H. te Riele, A. Timofeev, P. Zimmermann: *Factorization of a 768-bit RSA modulus*, 2010, http://eprint.iacr.org/2010/006.pdf