An Assessment of Attacks Strategies on the RSA Public-Key Cryptosystem

Jiri Just and John Coffey
Department of Computer Science
The University of West Florida
Pensacola, FL 32514

ABSTRACT

The first part of this paper briefly describes the history of RSA and the theory behind the scheme. The main part of this article provides an overview of RSA attack strategies which are grouped into major categories. Included are descriptions and assessments of factoring attacks, exponent-based attacks, forging RSA signatures, and hardware-based attacks. Each of the categories is analyzed for potential vulnerabilities that might be exploited. The paper concludes with a discussion of the overall security of this system.

1. INTRODUCTION

A fundamental requirement for Internet and other forms of networking is secure transmission of data. The sender must transmit data to the receiver without it being viewed or modified. The receiver requires assurance that the data was sent from the sender, not from someone using the sender's information. The objective of cryptography is information security, and the advent of public key cryptography was a significant step forward. One of the first major advances in public key cryptography was the RSA algorithm. It was the first algorithm known to be suitable for signing messages as well as for encryption, and is still widely used today for critical transmissions such as credit card transactions.

The first paper on the RSA was due to Ron Rivest, Adi Shamir and Leonard Adleman from MIT and was published in 1978 [1]. Impetus for their work came from a public-key cryptosystem conceived, but never implemented by Diffie and Hellman [2]. MIT was granted a patent for a communications system that used the algorithm in 1983 [1]. The algorithm is based upon public and private keys. The public key is made known and is used to encrypt a message so that anyone can do so. Only the recipient knows the private key which is needed to decrypt. The public key has two parts, the modulus n and the public exponent e. The private key has also modulus n and the private exponent d which is kept secret. The public and private keys are mathematical inverses of each other.

The remainder of this article explores the security of RSA and contains a survey of potential attacks on the algorithm. The attacks are categorized either by the basic approach taken, or by the part of the RSA system that the attack seeks to exploit (the message itself or the signature). The main attack categories, factoring, exponent based and hardware based, are described. The article closes by drawing some conclusions regarding vulnerabilities in the RSA cryptosystem.

2. FACTORING ATTACKS ON RSA

Factoring attacks are based upon the fact that the private key d can be computed if p and q can be discovered by factoring n. Given n and e (which is already known) d can be computed by solving the equation $de \equiv \mod \varphi(n)$ where the totient is $\varphi(n) = (p-1)(q-1)$. There are several approaches to the factoring of large numbers. The next sections will consider several in turn.

Brute-Force Factoring attack. This approach is based on searching for factors of n, p, and q by trying all possibilities. The size of the set of possible factors can be decreased by finding the square root of n and also by excluding even numbers and numbers ending in 5. Protection against the brute-force attack is to pick large primes p and q. Making p and q the same size also makes factoring n harder. For instance, in the current RSA competition, n is comprised of 205 digits. Therefore, brute force methods are computationally intractable.

General purpose factoring methods: General purpose factoring methods can be used to factor any numbers. One of the earliest general purpose factoring methods was the Sieve of Eratosthenes [3]. An algorithm called the general number field sieve [4] is the most efficient algorithm

known for factoring integers larger than 100 digits. An older method called the quadratic sieve method is also known. Both methods are based on the idea that one might factor n using the respective sieve to determine integers a, and b such that $a^2 \equiv b^2 \bmod n$ and $a \equiv (+-) b \bmod n$. Then, n divides $a^2 - b^2 = (a - b)(a + b)$, but neither a-b nor a+b. Hence, gcd ((a-b), n) is not a trivial factor of n. The general number field sieve or older quadratic sieve differ in the specific way the integers a and b satisfying $a^2 \equiv b^2 \bmod n$ and $a \equiv (+-)b \bmod n$ are found.

Crandall and Pomerance [5] present an algorithm that can factor a 70 digit number. According to Silverman [6], the algorithm will take about a day on modern computers. However, general purpose factoring methods can not factor any integer. General purpose factoring can be sped up by parallel computation which is based on giving each CPU a different set of polynomials. However, it is impractical to factor 200 digit numbers because doing so would require on the order of a billion times longer than 100-digit numbers according to Silverman [6]. Worse still, it is not currently possible to factor 100-digit numbers.

Special-purpose Factoring Methods: This type of factoring depends on the form of p and q. These methods are more efficient than general purpose ones if p and q are in the right format. Pollard's p-1 method [7] is one such example. The algorithm is based on the fact that numbers of the form $a^b - 1$ tend to be highly composite when b composite. Since it is computationally simple to evaluate numbers of this form in modular arithmetic, the algorithm makes it possible to check many potential factors quickly. In particular, the method will find a factor p if b is divisible by $p - 1$, hence the name. When $p - 1$ is smooth (the product of only small integers) then this algorithm is well-suited to the discovery of the factor p. However, several other constraints must be satisfied for the method to work. The main advantage of special-purpose factoring methods is that if the number is in the right form, it can be factored quite quickly**.**

Elliptic Curve Method. This method was introduced by Lenstra [7] and is one of the fastest factoring methods for numbers comprised of approximately 25 digits. It is an improved version of the Pollard p-1 method. The Lenstra elliptic curve factorization gets around the assumption that *n* has a prime factor *p* such that $p - 1$ had only "small" prime factors, by considering the group of a random elliptic curve (an algebraic curve defined by an equation of the form $y^2 = x^3 + ax + b$) over the finite field $\mathbf{Z}_p$, rather than considering the multiplicative group of $\mathbf{Z}_p$ which always has order *p*-1. As demonstrated in a theorem by Hasse the order of the group of an elliptic curve over $\mathbf{Z}_p$ varies between p+1-2*sqrt(p) and p+1+2*sqrt(p), which bounds the number of points on an elliptic curve over a finite field, above and below, and randomly, and is likely to be smooth for some elliptic curves. Elliptic curve factorization can be also executed on more than one processor. Each processor will get its own curve and will quit on first success. According to Silverman [6], a 38 digit number was factored by this method. A simple defense against this algorithm is to make n large and the factors the same size, since the algorithm starts with small factors first.

Factoring on a Quantum Computer. Shor's algorithm [8] is a polynomial-time integer factorization algorithm designed for implementation on quantum computers. Like many quantum computer algorithms, it is probabilistic: it gives the correct answer with high probability, and the probability of failure can be decreased by repeating the algorithm. However, since a proposed answer is verifiable in polynomial time, the algorithm can be modified to work both correctly and efficiently.

The algorithm consists of an iterative process of generating a random number a and computing `gcd(a,N)` If $a \neq 1$, the algorithm terminates with success. If not, a period-finding routine based in a quantum computer is called. The result of the quantum computation, r, is either a solution to the problem or additional iterations are performed. At present, it is difficult to state authoritatively if Shor's algorithm will be a threat to RSA or not because the period-finding subroutine must be tuned to each unique value of N and generally speaking, quantum computing is still much more an area of research than a scalable, deployable technology.

3. EXPONENT-BASED ATTACKS

To reduce encryption or signature-verification time, a small public exponent e is often used. The smallest possible value for e is 3. The following attacks are similar to factoring attacks in the sense that the goal is to find p, and q. However, low exponent attacks rely on a low public exponent in order to find the prime factors. The following attacks exploit low values of the exponent.

Wiener's Attack. In 1990, Wiener [9] observed that information encoded in the public exponent e might help to factor n. Wiener proposed an attack on the RSA system by a continued fraction approximation, using the public key (*n*, *e*) to provide sufficient information to recover the private key *d*. Wiener proved that if the keys in the RSA system are chosen such that $n = pq$, where $q < p < 2q$, and *d* <¼4*sqrt(n), then given the public key (*n*, *e*) with de= 1 mod φ the private key *d* can be computed in linear time.

This approach only works if d is chosen to be small relative to n. However some devices use small d because they have limited computational power. Wiener proposes certain techniques to avoid his attack such use of a large

encryption exponent. Boneh and Durfee [10] extended Wiener's work by showing that the attacker can efficiently compute d from (n, e) provided that $d < N^{0.292}$. They expect their algorithm to work on $d < N^{0.5}$. However, as stated in their conclusion, they can not state the approach to their attack as a theorem since they can not prove that it will always succeeds [10].

Small-Message Attack. RSA encryption is not effective if both the message m to be encrypted and the exponent e to be used for encryption are small relative to the modulus n. If $c=m^e < n$ is the cipher text, then m can be recovered from c by ordinary root extraction (the operation of taking an n^{th} radical root of a number). Therefore, either the public exponent should be large or the messages should always be large. Practically speaking, a small public exponent is often preferred with the message padded so that it is large, in order to speed up the encryption and to prevent Wiener's attack.

Low-Exponent Attack with Multiple Recipients. Hastad [11] showed that low exponent RSA is not secure if the same message is encrypted to several receivers. For example, let e = 3. Then if the number of receivers is 7, the eavesdropper can find the plaintext from the seven cipher texts of each receiver. If three senders participating in the same system encrypt the same message m using the same public exponent 3, the attacker can compute m from the three cipher texts even if the senders are using different modulo values: n_1, n_2, and n_3.

Generally, if k plain texts are encrypted with the same exponent e, an attacker can solve for m in polynomial time using lattice reduction techniques. This result has been extended by Coppersmith [12] who showed that RSA encryption with exponent 3 is vulnerable if the opponent knows two-thirds of the message, or if two messages agree over eight-ninths of their length. Low-Exponent attack is also a concern if the messages are related in a known way. Padding the messages with pseudorandom strings prior to encryption prevents mounting this attack in practice. If the messages are related in a known way, they should not be encrypted with many RSA keys. A recommended value of e that is commonly used today is $e=2^{16}+1$. One advantage of this value for e is that its binary expansion has only two ones, which implies that the square-and-multiply algorithm requires very few operations.

4. FORGING RSA SIGNATURES

A fairly significant real-world scare occurred because of an implementation flaw in error reporting during signature verification [18]. This vulnerability afforded the possibility of signatures being forged using only an RSA public key (without requiring the RSA private key). This problem included all RSA signatures.

The source of the problem was reportedly a failure of the implementation of the software that did not allow it to detect signatures which have been crafted to appear mostly valid. This failure to detect and alert on this category of signatures could create a situation where a forged signature may be trusted. The end result of a successful attack could include abusing trust relationships that have been established based on RSA keys or digital certificates, such as posing as a trusted party and signing a certificate or key. The vulnerability was detected and remediated before significant damage occurred.

5. ADAPTIVE CHOSEN CIPHERTEXT ATTACKS

In 1998, Bleichenbacher [13] described the first practical adaptive chosen ciphertext attack, against RSA-encrypted messages using the PKCS #1 v1 padding scheme (a padding scheme randomizes and adds structure to an RSA-encrypted message, so it is possible to determine whether a decrypted message is valid.) The Bleichenbacher sent millions of test ciphertexts to a decrypter in order to reveal the content of an RSA encrypted message. He showed that an RSA private-key operation can be performed if the attacker has access to an oracle that, for any chosen ciphertext, returns only one bit telling whether the ciphertext corresponds to some unknown block of data encrypted using PKCS #1. This method is especially dangerous for servers using SSL.

A recipient (server), may be vulnerable to this attack if it processes many messages, and reveals the success or failure of the operations. A protocol such as SSL may not require client authentication, so the attacker can easily remain anonymous through the process. In order to prevent adaptive-chosen-ciphertext attacks, it is necessary to use an encryption or encoding scheme that limits ciphertext malleability (transformations on the ciphertext to produce meaningful changes in the plaintext). Another countermeasure is to change the server's key pair frequently.

6. SUPERENCRYPTION

SuperEncryption is a process of running an already encrypted file through an encryption algorithm. A SuperEncryption attack against RSA was developed and reported by Simmons and Norris [14] shortly after RSA was published. It is based on the fact that a sufficient number of encryptions will eventually produce the original message. This result occurs because the RSA encryption function is mapped onto a finite set, which makes the graph of the function a union of disjoint cycles. This attack was a real threat to RSA while the number of encryptions required was small. However, if the communications use large primes which will be used in random, superencryption becomes impractical.

7. HARDWARE-BASED ATTACKS

The next two attacks can be categorized as hardware based because they exploit knowledge of the sender's or recipient's hardware.

Timing attacks. In 1995, Kocher[15] described a timing attack on RSA based on knowing specific attributes of the recipient's hardware. If the attacker has ciphertext from the sender, the ciphertext can be used to determine the private key by measuring decryption times on the recipient. This attack can be used either to determine d or to attack the RSA signature. In 2003, Boneh and Brumley[16] demonstrated a different timing attack that exploits data recovered from a Secure Socket Layer (SSL)-enabled server to attain an RSA factorization. This attack takes advantage of information leaked by the Chinese remainder theorem optimization used by many RSA implementations. The most effective defense is to arrange to have the same decryption times for different ciphertext values.

Branch Prediction Analysis (BPA) attacks. Many processors use a Branch predictor to determine whether a conditional branch in the instruction flow of a program is likely to be taken or not. Usually these processors also implement Simultaneous multithreading (SMT). Branch Prediction Analysis attacks use a spy process running in parallel on the same processor as a process executing an RSA algorithm to discover the private key. Simple Branch Prediction Analysis (SBPA) claims to improve BPA in a non-statistical way. In [17], the authors of SBPA claim to have discovered 508 out of 512 bits of an RSA key in a single signing operation. Their attack was against an OpenSSL RSA implementation. They conclude that memory protection, sandboxing and virtualization, fail to prevent such "side channel" attacks and that these attacks are much more dangerous than "pure" timing attacks.

8. CONCLUSIONS

While this article presents a snapshot in time, the question of vulnerabilities in the RSA system is an ongoing one. As an example, it is now feasible to factor keys of the lengths used in the earliest specifications of the system. The simple remedy was to make keys longer. However, some current vulnerabilities might be identified. Clearly, the exponent must be larger than 3 to decrease the possibility of low exponent attacks. Specific implementations might introduce vulnerabilities that do not stem from the specification, as was the case with the signature forgery attacks. In the case of servers, the key should be changed often to prevent adaptive chosen ciphertext attacks**.** Otherwise, RSA appears to contain few vulnerabilities that can be exploited However, with advances in quantum computers, all bets may be off.

In summary, it is possible that the main area of vulnerability in this scheme pertains to the human beings who create and use implementations of it. RSA software comes with default settings which differ from company to company. Most implementations are set up to provide basic security, which means that the exponent used can be low and can allow the attacker to use one of the low exponent attacks. The key can be set to 64 bits which is easy to break using some factoring methods. New installations must always be checked for these settings to ensure security. It is worth noting that as computing methods evolve, cryptology will also advance, but so will the sophistication of attacks. Consequently, the security of RSA will remain a matter requiring ongoing vigilance.

9. REFERENCES

[1] R. Rivest, A. Shamir, and L. Adleman, A method for obtaining digital signatures and public-key cryptosystems. **Communications of the ACM,** Volume 21 , No. 2. 1978, pp. 120-126.

[2] W. Diffie and M. Hellman, New directions in cryptography, **IEEE Transactions on Information Theory**, 6. 1976, pp 644-654.

[3] P. Pritchard, Linear prime-number sieves: a family tree. **Scientific Computer Programming** Volume 9, No. 1, 1987, pp 17-35.

[4] J.P. Buhler, H.W. Lenstra, and C. Pomerance, **The development of the number field sieve,** Volume 1554 of Lecture Notes in Computer Science, Springer-Verlag, 1994.

[5] R. Crandall and C. Pomerance **,Prime Numbers: A Computational Perspective,** Springer. ISBN 0-387-94777-9, 2001, pp.227–244.

[6] R.D. Silverman, Massively distributed computing and factoring large integers, **Communications of the ACM,** Volume 34 No. 11, pp. 242-299.

[7] H.W. Lenstra Jr, Factoring integers with elliptic curves, **Annals of Mathematics**. 1987, pp. 649-673.

[8] P.W. Shor, Algorithms for quantum computation: Discrete logarithms and factoring, **35th Annual IEEE Symposium on the Foundations of Computer Science** (1994), pp. 124-134.

[9] M.J. Wiener, Cryptanalysis of Short RSA Secret Exponents, **IEEE Transactions on Information Theory,** Volume 36, No. 3, 1990, pp. 553-558.

[10] D. Boneh, and G. Durfee, Cryptanalysis of RSA with private key d less than $N^{0.292}$. **IEEE Transactions on Information Theory IT.** Volume 46. No. pp.

[11] J. Hastad, Solving simultaneous modular equations of low degree. **SIAM Journal of Computing,** Volume 17, No. 2, 1988, pp. 336–341.

[12] D. Coppersmith, Small solutions to polynomial equations, and low exponent RSA vulnerabilities. **Journal of Cryptography,** Volume 10, No. 4, 1997, pp. 233–260.

[13] D. Bleichenbacher, Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1, **Cryptology** 1998, pp. 1–12.

[14] G. Simmons and M. Norris, Preliminary comments on the MIT public-key, **Cryptologia,** Volume 1, No. 4, 1977, pp. 406-414

[15] P.C. Kocher, Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems, Online, Available: **http://www.cryptography.com/ resources/ whitepapers/TimingAttacks.pdf.**

[16] Boneh and Brumley, Remote Timing Attacks are Practical, Online, Available: http://crypto. stanford.edu/~dabo/papers/ssl-timing.pdf.

[17] O. Aciicmez, C. K. Koc and J-P Seifert, On the Power of Simple Branch Prediction Analysis, **Cryptology,** ePrint Archive, Report 2006/351

[18] Juniper Networks Multiple Vendor SSH RSA Signature Forging Vulnerability. Online, Available: http://www.juniper.net/security/auto/vulnerabilities/ vuln8094.html