Research Project: Attacks on RSA

Hongyan Wang - For PMAT 669

December 8, 2014

Department of Mathematics and Statistics

1 Introduction

1.1 Description of RSA

The RSA cryptosystem, developed in 1977 by Ron Rivest, Adi Shamir and Len Adleman, is one of the first practicable public-key cryptosystems. It is commonly used to ensure privacy and authenticity of digital data. RSA is based on the practical difficulty of factoring the product of two large prime numbers. First we briefly describe the RSA encryption and decryption algorithm.

Pick two random large primes p and q. Set $N = pq$ and $\Phi(N) = (p-1)(q-1)$. Then select a random integer $e \in \mathbb{Z}^*_{\Phi(N)}$ and compute $d \in \mathbb{Z}^*_{\Phi(N)}$ such that $de \equiv 1 \bmod \mathbb{Z}^*_{\Phi(N)}$. We call N is the modulus of this RSA system, e is the public exponent(encryption key) and d is the private exponent (decryption key). We publish (e, N) and keep d secret. The sender uses the receiver's public key to encrypt the message and the receiver decrypts the encrypted message by the secret key.

For example, if Alice wants to send the message $M \in \mathbb{Z}^*_N$ to Bob, she computes

$$C \equiv M^e \ (\bmod N)$$

and sends C to Bob. When Bob receives the ciphertext C, he can recover the message by computing

$$M \equiv C^d \ (\bmod N).$$

RSA decryption algorithm is a one-way trapdoor function. If we know the trapdoor, private key d, we can easily break RSA system. Besides, recovering d is equivalent to factor the modulus N. Thus, one can break RSA cryptosystem by factoring N while others may find ways to recover the message M without knowing d or the factorization of N.

Since RSA was invented, it has been analyzed for vulnerability by many researchers and has faced lots of attacks. Although none of these attacks is devastating, it may be dangerous because of improper use of the system. Our goal is to examine some of these attacks and analyze the underlying mathematical tools they use. Roughly, we can categorize these attacks into two categories. One is mathematical attacks, which concentrate on the analysis of structures of RSA cryptosystem. The other is implementation attacks, which focus on the implementation of RSA.

1.2 This paper

In this paper, we will first talk about mathematical attacks. Some simple elementary attacks, like factorization attack and common modulus attack will be discussed. We will also present some attacks based on the small public exponent, like broadcast attack and related message attack. Besides the attack based on the small private exponent, like Wiener's continued fractions attack will also be discussed. At last, we will introduce several implementation attacks, like timing attack and Bleichenbacher's CCA attack. Furthermore, we will discuss two attacks in detail, one is Wiener's continued fractions attack and the other is Bleichenbacher's CCA attack.

2 Elementary Attacks

2.1 Integer Factorization Attack

It is well-known that breaking RSA is equivalent to factoring large integers. So the most direct method of breaking RSA is to factor the RSA modulus N. The integer factoring algorithms can be categorized into two categories: special purpose algorithms and general purpose algorithms. The efficiency of special purpose algorithms depends on the unknown factors while the efficiency of general purpose algorithms depends on the integer we want to factor. Usually, the special purpose algorithm is the perfect choice for small integers. However, in RSA system we always use large modulus. Thus, general purpose algorithms play a more important role in the factorization of N.

One special purpose factorization algorithm is Fermat's Factorization Method. It tries to find two integers x and y such that $N = x^2 - y^2$. Then $N = (x + y)(x - y)$, so $(x + y)$ and $(x - y)$ are probably the nontrivial factor of N.

Some efficient general purpose algorithms, like Quadratic Sieve, Continued Fraction Method, Shanks' Square Form Factorization and Number Field Sieve, improve Fermat's algorithm by finding two integers x and y such that

$$x^2 \equiv y^2 \ (\bmod N), 0 < x < y < N, x + y \neq N.$$

Then gcd $(x \pm y,\ N)$ are possibly the nontrivial factor of N, since $N|(x+y)(x-y)$, but $N \nmid (x+y)$ and $N \nmid (x-y)$. The main task is how to find such x and y. Different methods are used in these four algorithms. Nowadays, the most efficiently one is general number field sieve(GNFS). The complexity for factoring N using GNFS is

$$\mathcal{O}(\exp((\sqrt[3]{\frac{64}{9}} + o(1))(\ln N)^{\frac{1}{3}}(\ln \ln N)^{\frac{2}{3}})).$$

People have used GNFS to factor RSA-768(768 bits N), the largest factored RSA number. However, N is typically 1024 or 2048 bits long. Thus, we can still presume that RSA is secure.

2.2 Blinding Attack on RSA Signature

The blinding attack enables the attacker, Eve, to obtain a valid signature on a message of her choice by asking Alice to sign a random "blinded" message. Assume that Alice has the private key d and public key $(e,\ N)$. Imagine the case when Eve wants Alice's signature S on a message $M \in \mathbb{Z}_{\mathbb{N}}^*$. However, Alice refuses to sign on this message. Then Eve picks a random integer $r \in \mathbb{Z}_{\mathbb{N}}^*$ and sets $M' = r^e M \pmod N$. In this case, Alice may be willing to provide the signature S' on M'. Then

$$S' \equiv (M')^d \equiv (r^e M)^d \equiv r^{de} M^d \equiv rM^d \equiv rS \pmod N.$$

Thus, Eve can easily get the signature S by computing $S = S'/r$. And Alice has no idea what message she's signings.

2.3 Common Modulus Attack

The common modulus attack shows that using the common modulus to encrypt the same message is dangerous. Two cases are discussed here.

One case is that the trusted third party provides the same modulus N for two users. If Alice and Bob both use the modulus N and their public keys are $(N,\ e_1)$ and $(N,\ e_2)$ respectively, then Bob can use his own pair of keys $(e_2,\ d_2)$ to factor N. Hence, Bob may be able to recover Alice's private key from her public key. Thus, an RSA modulus can never be used by more than one party.

The other case is, Alice uses e_1 and e_2 to encrypt the same message M. Then Eve can use the ciphertext to recover the message without knowing the trapdoor d. Because the probability that $e_1$and e_2 are relatively prime is really high. So Eve can use Extended Euclidean Algorithm to find integers x and y such that $xe_1 + ye_2 = 1$. If

$$C_1 \equiv M^{e_1} (\text{mod } N)\ ,\ C_2 \equiv M^{e_2} (\text{mod } N),$$

then

$$C_1^x \cdot C_2^y \equiv M^{xe_1} \cdot M^{ye_2} \equiv M^{xe_1 + ye_2} \equiv M \ (\text{mod } N).$$

So the secret message can be recovered efficiently.

2.4 Fixed-Point Attack

The fixed-point attack was discovered by Simmons and Norris as soon as the invention of RSA in 1977[6]. They intended to recover the message M without knowing the factorization of N or the trapdoor information.

Definition 2.4.1 (The fixed-point) Let $0 \leq x < N$. If we can find a positive integer k such that $x^{e^k} \equiv x \pmod{N}$, then x is called a fixed-point of RSA (e, N) and k is the order of this fixed-point.

Theorem 2.4.2 If C is the fixed-point of RSA (e, N) with order k, i.e. $C^{e^k} \equiv C \pmod{N}$, then

$$C^{e^{k-1}} \equiv M \pmod{N}.$$

To recover the message of an encryption C, we just need to compute the sequence of C^{e^1}, C^{e^2},... modulo N. As soon as we find an integer k such that $C^{e^k} \equiv C \pmod{N}$, we stop the computation and obtain the message by computing $M \equiv C^{e^{k-1}} \pmod{N}$.

Apparently, if e has the order r in the multiplicative group modulo $\Phi(N)$, i.e. r is the smallest integer such that $e^r \equiv 1 \pmod{\Phi(N)}$, then we need r times computation. Thus, to prevent RSA from this attack, we need to make sure that r is large enough. However, in practical application, the probability that the random e has a large order is really high. Furthermore, it has been proved that if this attack works, we can factor N. It means this attack does no better than factoring the modulus N[9]. Thus, RSA is invulnerable to the fixed-point attack.

3 Attacks Based on the Small Public Exponent

3.1 Hastad's Broadcast Attack

The broadcast attack is based on the broadcast of the same message to different parties. The attack enables the attacker to recover the message using the public information of these parties.

To broadcast a message M to k different parties P_1, P_2,..., P_k, Bob needs to use each party's public key e_i to encrypt the message and then send the encryption C_i (mod N_i) to P_i. Consider the simplest case, all the parties use the same public keys 3. Then if Eve obtains C_1, C_2, C_3, she can recover the message M efficiently. We may assume that gcd (N_i, N_j) = 1 for $i \neq j$ and $M < N_i$ for any $1 \leq i \leq k$. Since

$$C_1 \equiv M^3 \pmod{N_1},\ C_2 \equiv M^3 \pmod{N_2},\ C_3 \equiv M^3 \pmod{N_3},$$

Eve may use Chinese Remainder Theorem to find C' such that $C' \equiv M^3 \pmod{N_1N_2N_3}$. Because $M < N_i$, $M^3 < N_1N_2N_3$. Then Eve just needs to find the real cube root of C' to find M.

Based on that, Bob may want to pad the message before the encryption. However, simple linear padding is not secure. Furthermore, Hastad showed that applying any fixed polynomial to the message before the encryption is not helpful to prevent the attack.

Theorem 3.11 (Hastad) Let N_1, N_2,..., N_k be pairwise relatively prime integers, $N_{min} = min_i(N_i)$, $N = N_1N_2 \dots N_k$ and $f_i(x) \in \mathbb{Z}_{N_i}[x]$. For a plaintext $M < N_{min}$, if $k \geq max_i\{e_i deg(f_i(x))\}$, then given $c_i = f_i(M) \pmod{N_i}$ and $\langle N_i, e_i \rangle_{i=1}^{k}$, one can compute the plaintext M in time polynomial in $\log N$ and $max_i\{e_i deg(F_i(x))\}$.

As a result, to prevent the broadcast attack, we need to avoid broadcasting the same message to large number of parties. It is also necessary to use large public exponents.

3.2 Franklin-Reiter Related Message Attack

The related message attack depends on the related messages one sends to the other. Franklin and Reiter proved that the RSA system with low public exponent is vulnerable to this attack. They showed that when Alice sends two related messages, M and $aM + b$, to Bob using the same modulus N and small public exponent $e = 3$, one can recover the message M efficiently without knowing the factorization of N.

Assume that $C_1 = M^3 \pmod{N}$ and $C_2 = (aM + b)^3 \pmod{N}$. Then given $\langle C_1, C_2, N, a, b \rangle$, one can recover the secret message M as follows:

$$M \equiv \frac{3a^3bM^2 + 3a^2b^2M + 3ab^3}{3a^3bM^2 + 3a^2b^2M + 3ab^3} M \equiv \frac{b(C_2 + 2a^2C_1 - b^3)}{a(C_2 - a^3C_1 + 2b^3)} \pmod{N}.$$

After that, they also showed that for arbitrary e, there is a simple way to recover M.

Theorem 3.2.1 Let $C_1 = M^e \pmod{N}$ and $C_2 = (aM + b)^e \pmod{N}$. Then compute the gcd $(x^e - C_1, (ax+b)^e - C_2) \in \mathbb{Z}_N[x]$, which will yield the linear polynomial $x - M$ except in rare cases.

It has been proved that the gcd of these two polynomials can be computed in time $\mathcal{O}((elog(e))^2)$. Thus, the attack may be practical for all exponent e with length up to 32 bits. For example, this attack may be very efficient against $e = 2^{16} + 1$, which is a popular choice in many applications.

3.3 Coppersmith's Short Pad Attack

The short pad attack is applied in the case when Bob sends Alice a message M using a small random padding prior to the encryption. This attack allows Eve to recover the message M without knowing the factorization of N or the trapdoor information.

Assume Alice's public key is $\langle e, N\rangle$ where N is n-bits long. Let $m = \left\lfloor \frac{n}{e^2} \right\rfloor$ and $M \in \mathbb{Z}_N^*$ is the message with the length at most $n - m$ bits. When Bob tries to send a message M to Alice, he performs some short padding before the encryption, say $M_1 = 2^m M + r_1$ and $C_1 = M_1^e$. If Eve prevents the message from reaching the destination, then Bob has to send the message M again to Alice because he does not get the respond. Again, Bob performs a random padding, $M_2 = 2^m M + r_2$ and $C_2 = M_2^e$. r_1 and r_2 are distinct integers with $0 < r_1, r_2 < 2^m$. In this case, given $\langle e, N, M_1, M_2, C_1, C_2\rangle$, Eve can recover the message M efficiently without knowing the padding r_1 and r_2.

Let $g_1(x, y) = x^e - C_1$ and $g_2(x, y) = (x+y)^e - C_2$. Apparently, if $y = r_2 - r_1$, $x = M$ is the common root of these two polynomials. Thus, as long as Eve knows $r_2 - r_1$, she can use Franklin-Reiter related message attack to find M_1 and M_2. Since $r = r_2 - r_1$ is the root of resultant $h(y) = res_x(g_1, g_2) \in \mathbb{Z}_N[y]$, according to Coppersmith's theorem[3], r can be found efficiently. Because the degree of $h(y)$ is at most e^2 and $|r| < 2^m < N^{1/e^2}$ is a small root modulo N. Once Eve knows $r_2 - r_1$, she is able to recover M.

3.4 Partial Key Exposure Attack

The partial key exposure attacks says that if an attacker knows $\frac{n}{4}$ least or most significant bits of private key d, where n is the length of modulus N, then the attacker can recover d in time $\mathcal{O}(elog(e))$. In this paper we just discuss the case when the attacker knows $\frac{n}{4}$ least significant bits of d.

Apparently, there exits an integer k such that

$$ed - k\Phi(n) = ed - k(N - p - q + 1) = 1.$$

Since $d < \Phi(n)$, then $0 < k \leq e$. Plug $q = \frac{N}{p}$ in the equation above and reduce it modulo $2^{n/4}$, then

$$p(ed) - kp(N - p + 1) + kN \equiv p \ (\bmod 2^{n/4}).$$

The attacker Eve knows e and $\frac{n}{4}$ least significant bits of d, then she knows ed mod $2^{n/4}$. It means she can get an equation in p and k. For each possible $k \leq e$, she solves the equation and gets the candidate values for p modulo $2^{n/4}$. Then for each p she tries to factor N. It has been shown that there are at most $elog(e)$ possible p modulo $2^{n/4}$. Thus, after at most $elog(e)$ attempts, N can be factored.

To defend partial key exposure attack, we need to keep entire bits of private key d secure. It is also very crucial to avoid the use of small public exponent.

4 Attacks based on the Small Private Exponent

4.1 Wiener's Continued Fractions Attack

In the real application of RSA, we may use a small private exponent to speed up the decryption. However, sometimes it is not a wise choice. In 1990, Wiener described an attack on RSA based on the small private exponent d. He showed that given (e, N), if $d < \frac{1}{3}N^{\frac{1}{4}}$ then d can be recovered efficiently using the continued fraction algorithm.

4.1.1 Introduction to Continued Fractions

Definition 4.1.1 (Continued Fractions) A continued fraction is an expression of the form

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{\cdots}}}}$$

with a_i are all integers, where $a_i > 0$ for $i \geq 1$ and $a_0 \geq 0$.

For convenience, we write a continued fraction as $r = [a_0, a_1, a_2, \ldots]$.

Definition 4.1.2 (Partial Convergents) For $0 \leq n \leq m$, the nth convergent of a continued fraction $[a_0, a_1, \ldots,]$ is $c_n = [a_0, a_1, \ldots, a_n]$. These convergents for $n < m$ are also called partial convergents.

For $-2 \leq n \leq m$, we define non-negative integers p_n and q_n as follows:

$$p_{-2} = 0,\ p_{-1} = 1,\ p_0 = a_0,\ \cdots\ p_n = a_n p_{n-1} + p_{n-2},\ \cdots,$$
$$q_{-2} = 1,\ q_{-1} = 0,\ q_0 = 1,\ \cdots\ q_n = a_n q_{n-1} + q_{n-2},\ \cdots.$$

Theorem 4.1.3 (Partial Convergents) For $0 \leq n \leq m$, we have $c_n = [a_o, a_1, \ldots, a_n] = \frac{p_n}{q_n}$.

It has been proved that every real number can be represented by a continued fraction.

Theorem 4.1.4 (Approximation) Every irreducible rational fraction $\frac{a}{b}$ that satisfies the inequality

$$| \alpha - \frac{a}{b} | < \frac{1}{2b^2}$$

is a partial convergent of the number α.

These two theorems are the foundation of Wiener's attack. The main idea of this attack is to expand $\frac{e}{N}$ as a continued fraction while d is the denominator of one partial convergent of the expansion.

4.1.2 Attacks on RSA

Theorem 4.1.5 Let $N = pq$ with $p < q < 2p$. If $d < \frac{1}{3}N^{\frac{1}{4}}$, d is the denominator of a partial convergent of the continued fraction expansion of $\frac{e}{N}$.

Proof: Since $de \equiv 1 \pmod{\Phi(N)}$, there exists k such that $de - k\Phi(N) = 1$. $\Phi(N) = (p-1)(q-1) = N - (p+q) + 1$ and $p < q < 2p$ imply that $n - 3\sqrt{N} < \Phi(N) < N$. Thus,

$$| \frac{k}{d} - \frac{e}{N} | = | \frac{kN - de}{dN} | = | \frac{kN - k\Phi(N)}{dN} | < \frac{3k}{d\sqrt{N}}$$

Because $e < \Phi(N)$ and $de - k\Phi(N) = 1$, we have $k < d$. $d < \frac{1}{3}N^{\frac{1}{4}}$ implies that $\frac{3}{\sqrt{N}} < \frac{1}{2d^2}$. Then,

$$\frac{3k}{d\sqrt{N}} < \frac{3d}{d\sqrt{N}} = \frac{3}{\sqrt{N}} < \frac{1}{2d^2},$$

which means $| \frac{k}{d} - \frac{e}{N} | < \frac{1}{2d^2}$. According to theorem 4.1.4, $\frac{k}{d}$ is a partial convergent of $\frac{e}{N}$. It indicates that d is the denominator of a partial convergent of the continued fraction expansion of $\frac{e}{N}$.□

Because the partial convergents of the continued fraction expansion of a real number satisfy

$$q_n = a_n q_{n-1} + q_{n-2},$$

the denominators grow exponentially. It means the total partial convergents of $\frac{e}{N}$ is of order $\mathcal{O}(log(N))$.

Wiener also described a method to tell if a partial convergent,$\frac{a}{b}$, of the expansion of $\frac{e}{N}$ is exactly $\frac{k}{d}$. If $\frac{a}{b} = \frac{k}{d}$, then $\Phi(N) = \frac{ed-1}{k} = \frac{eb-1}{a}$. Thus, we can factor N with the knowledge of $\Phi(N)$ and N. For one convergent $\frac{a}{b}$, if we obtain a factorization of N, it is reasonable to claim that $\frac{a}{b} = \frac{k}{d}$.

Another method to test each convergent is to pick a random $M \in \mathbb{Z}_N^*$, compute M^{eb} mod N and see whether or not $M \equiv M^{eb} \bmod N$.

4.1.3 Defense of the Attack

It is straightforward that to defend this attack, we need to make sure $d > \frac{1}{3}N^{\frac{1}{4}}$. However, it may compromise the speed of decryption algorithm. Wiener described some techniques that ensure the fast decryption while at the same are not vulnerable to his attack.

One method is to use large public exponent e. Instead of reducing e mod $\Phi(N)$, we use $e' = e + t\Phi(N)$ where t is a large integer. Apparently, we can still encrypt message using e'. However, when e' is large, k is also very large. It has been showed that when $e' > N^{1.5}$ this attack does not work no matter how small d is. However, at the same time, when e is pretty large, it will increase the time for encryption.

Another way is to use CRT. Details are presented as follows:

1. Compute

   $$M_p = C^{dp} \ (\bmod p) \text{ and } M_q = C^{dq} \ (\bmod q),$$

   where $d_p \equiv d \ (\bmod p-1)$ and $d_q \equiv d \ (\bmod q-1)$.

2. Use the EEA to solve the linear congruences

   $$qq^* \equiv 1 \ (\bmod p) \text{ and } pp^* \equiv 1 \ (\bmod q)$$

   for $0 < p^* < q$ and $0 < q^* < p$.

3. Set $M \equiv qq^* M_p + pp^* M_q \ (\bmod N)$.

Although d_p and d_q may be small, the value of d mod $\Phi(N)$ can be large. Thus, the attack will not work in this case.

In 2002, Durfee and Boneh discovered a new attack that improved the bound for d slightly. They showed that as long as $d < N^{0.292}$, the attacker can recover d efficiently using (e, N). Though this attack works well in practice, they failed to find the proof. It is likely that the bound of d should be $N^{0.5}$. However, it is still an open problem.

5 Implementation Attacks

5.1 Timing Attack

The timing attack is based on the server's respond time to decryption queries. It was put forward by Kocher in 1995. He measured the hardware's respond time to perform an RSA signature or a decryption using the modular exponentiation algorithm. Then he found that the attacker was able to recover the decryption key without factoring N.

Consider the case when the hardware computes $C = M^d$ mod N. First, it translates d into binary form, i.e. $d = d_k d_{k-1} ... d_2 d_1 d_0$ where $d_i \in \{0, 1\}$ for $i = 0, 1, ..., k$. The exponentiation algorithm can be described as follows:

1. Set $M_0 = M$ and $C = 1$.
2. For $i = 0, 1, ..., k$, if $d_i = 1$, $C = C \cdot M_0$ mod N.
3. Set $M_0 = M_0^2$ mod N.

After k steps, $C = M^d$ mod N. During the attack, Eve asks the hardware to compute $C_i = M^d$ mod N for a large number of M_i and records the time T_i it takes for each computation. Since d has to be odd, so $d_0 = 1$. In the second iteration, $M_0 \equiv M^2$ mod N and $C = M$. If $d_1 = 1$, it computes $C = M_0 \cdot C = M^2 \cdot M$ mod N. Otherwise, it does not. Use t_i to denote the time it takes to compute $M_i^2 M_i$ mod N. Since t_i's depend on the value of M_i's, they are different from each other. Kocher observed that when $d_1 = 1$, $\{T_i\}$ and $\{t_i\}$ are correlated. When $d_1 = 0$, $\{T_i\}$ and $\{t_i\}$ behave as independent random variables. By using the knowledge of probability and statistics to measure the correlation, Eve can determine whether $d_1 = 1$ or $d_1 = 0$.

Similarly, Eve can recover d_2, d_3, ... If a small public key is used, Eve just needs to obtain a quarter of the bits of d and then applies partial key exposure attack to recover d.

Usually, there are two simple ways to defend this attack. One is to add a delay so that the exponentiation algorithm always takes the fixed amount of time. The other one is based on the blinding. Before the signature, the server picks a random $r \in \mathbb{Z}_N^*$ and computes $M' = Mr^e \pmod N$. Then it signs M' and gets $C' = rC$. After that it sets $C = C'/r$. In this way, the server applies d to a random message M' which is unknown to the attacker. Thus, this attack will not work.

5.2 Random Faults Attack on RSA Signature

The random faults attack is based on the hardware or software error. It can be used to factor N and then break the RSA system completely. The Chinese Remainder Theorem(CRT) is used frequently to speed up the computation $M^d \bmod N$. Instead of working on modulo N, the signer first signs the message modulo p and q then combine the result using CRT. Detailed has been presented above in section 4.1.3.

When Alice uses CRT method to generate her signature, it is dangerous if there exits a glitch on her computer, which will cause a miscalculation. Assume C_p is computed incorrectly but C_q is correct. Then the resulting signature will be $C' \equiv qq^*C'_p + pp^*C_q \pmod N$. Since $C'^e \neq M \pmod N$, the attacker knows it is an invalid signature once he receives it. It is easy to show that $C'^e \neq M \pmod p$ while $C'^e = M \pmod q$. Thus, gcd $(N, C'^e - M)$ is a nontrivial factor of N.

In this attack, we also assume that Alice does not pad the message before she signs it. Random padding prior to signing will defeat this attack since the attacker cannot get the full knowledge of M. To defend this attack, it is necessary for Alice to check her signature before she sends it out.

5.3 Bleichenbacher's CCA Attack on PKCS #1 v1.5

5.3.1 PKCS #1 v1.5

The basic idea of PKCS #1 v1.5 is to pad a message before the encryption. Assume that the byte-length of modulus N is n, thus, $2^{8(n-1)} < N < 2^{8n}$. Suppose the byte-length of a message M is m and here m must not exceed $n - 11$. The message is encrypted as follows:

1. Choose a random padding string PS with byte-length $n - 3 - m$, such that PS contains no 00-byte. So the byte length of PS is at least 8.

2. Set $M' := 00||02||PS||00||M$. Interpret M' as an integer such that $0 < M' < N$.

3. Compute the ciphertext $C = M'^e \pmod N$.

So the first two byte of the padding are constant and the third one varies. After the padding, the message M' looks like this:

00	02	Padding String	00	Original message M

In this way, we can get a message with n byte-length. A message block in this form is called PKCS conformant. We also call a ciphertext C PKCS conformant if its decryption is PKCS conformant.

To decrypt, we first compute $M' = C^d \pmod N$ and interpret M' as a bit string. If M' is PKCS conformant, i.e., $M' = 00||02||PS||00||M$, then it is easy to find original message M.

5.3.2 Description of the Attack

In this attack, we assume that the attacker has access to an oracle $\mathcal{O}$, which can tell whether or not a ciphertext is conformant. This is the only prerequisite. When we input a ciphertext C, the oracle outputs 1 if C is conformant, or 0 if not.

The following is the rough idea of this attack, details can be found in[11]:

First, we need to set $B = 2^{8(n-2)}$.

1. The attacker intercepts a ciphertext $C = M^e \pmod N$. If we assume C is conformant, then $M \in [2B, 3B)$.
2. Pick a small integer s and compute $C' = Cs^e \pmod N$.
3. Use the oracle to determine whether or not $C'$is conformant.
4. If so, $2B \leq Ms - rN < 3B$ for some r, then $\dfrac{2B + rN}{s} \leq M < \dfrac{3B + rN}{s}$.
5. By choosing different s, the attacker can reduce the possible solutions until only one left, i.e. $a \leq M < a + 1$. Then $M = a$.

5.3.3 Analysis of the Attack

Bleichenbacher showed that the expected number of chosen ciphertexts we need to recover the original message M is around

$$3/\Pr(P) + 16n/\Pr(P|A),$$

where $0.18 \times 2^{-16} < \Pr(P) < 0.97 \times 2^{-8}$ and $\Pr(P|A) = (\dfrac{255}{256})^8 \cdot (1 - (\dfrac{255}{256})^{n-10}$.

For example, when $n = 128$ (N is 1024 bits), this attack only needs about 2^{20} chosen ciphertexts to succeed.

After Bleichenbacher put forward this attack, he applied it to the SSL v3.0 protocol. Based on different error messages returned from the SSL server, Bleichenbacher was able to tell if the chosen ciphertext was conformant. Hence he used SSL server as the decryption oracle to decrypt the encrypted PreMasterSecret, which was used to generate SSL session keys.

Based on Bleichenbacher's chosen ciphertext attack on PKCS #1 v1.5, people then suggested PKCS #2, which prevent this attack by using OAEP. The main idea of OAEP is to use the hash function and random oracles to construct a Feistel network. OAEP makes it really improbable that a random string matches the padding format. Thus, it prevents Bleichenbacher's attack. When OAEP is implemented with certain trapdoor permutations, like RSA, we can prove it is secure against the chosen ciphertext attack[12].

5.3.4 Access to an Oracle

The only prerequisite for this attack is the access to an oracle. Such an oracle may be given in many practical scenarios. In this section, we just describe two situations.

The first one is based on the difference between the error message for the decryption and the error message for the signature. We know the integrity check is always an important part of the encryption. The receiver need to check both the message and the signature when he receives the encryption. He will accept the message only when the message is conformant and the signature is correct. An attacker does not have the ability to create a correct signature. However, if the the receiver gives detailed information about the error, then the attacker may be able to determine whether the chosen ciphertext is conformant or not. Because there is difference between the error information for the message that is not conformant and the error information for the message where only the signature is wrong.

The other case is a timing attack. In the application of RSA, we always combine the signature and the encryption. Consider the case that the receiver verifies the message in these two steps:

1. Decrypt the encrypted message. Reject if it's not conformant.

2. It it's conformant, check the signature. Reject if it's incorrect.

Since the attacker does not have the ability to generate the correct signature, the chosen ciphertext will definitely be rejected. If the encryption is conformant, the signature will be checked after that. Thus, it will take more time to receive the respond from the server. In the worst case, to check the signature is much more time consuming than the check for the decryption. It is much easier for the attacker to determine whether or not the chosen ciphertext is conformant.

6 Conclusion

Since RSA was invented in 1977, it has withstood the test of time and much research and is still proved to be secure. Although there are lots of attacks on RSA, there are also many anti-attacks against these attacks, which make RSA more secure. It has been shown that a quantum computer has a polynomial time algorithm for factoring

integers, however, it still has a long way to engineer a quantum computer [14]. Thus, RSA is considered to be safe for now. Improper implementations of RSA have been shown to be insecure. We just need to avoid making such mistakes.

References

[1] Menezes, Alfred; van Oorschot, Paul C.; and Vanstone, Scott A. Handbook of Applied Cryptography. CRC Press, October 1996.

[2] Boneh D. Twenty years of Attacks on the RSA Cryptosystem [J]. Notices of the AMS, 1999, 46(2): 203-213.

[3] Yan S Y. Cryptanalytic attacks on RSA [M]. Springer, 2007.

[4] da Costa Boucinha F. A Survey of Cryptanalytic Attacks on RSA [J]. 2011.

[5] Salah I K, Darwish A, Oqeili S. Mathematical attacks on RSA cryptosystem [J]. Journal of Computer science, 2006, 2(8): 665.

[6] Simmons, G. J. and Norris, M. J. Preliminary Comments on the MIT Public-Key Cryptosystem. Cryptologia 1, 406-414, 1977.

[7] Joye M, Lepoint T. Partial key exposure on RSA with private exponents larger than N [M]//Information Security Practice and Experience. Springer Berlin Heidelberg, 2012: 369-380.

[8] Wiener M J. Cryptanalysis of short RSA secret exponents [J]. Information Theory, IEEE Transactions on, 1990, 36(3): 553-558.

[9] Dujella A. Continued fractions and RSA with small secret exponent [J]. arXiv preprint cs/0402052, 2004.

[10] BA Cipra; Safe against cycling: researchers confirm invulnerability of RSA cryptosystem. SIAM News, 34 (2001).

[11] Jager T, Schinzel S, Somorovsky J. Bleichenbacher's attack strikes again: breaking PKCS# 1 v1. 5 in XML Encryption [M]//Computer Security–ESORICS 2012. Springer Berlin Heidelberg, 2012: 752-769.

[12] Bleichenbacher D. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS# 1 [C]//Advances in Cryptology—CRYPTO'98. Springer Berlin Heidelberg, 1998: 1-12.

[13] Bellare M, Rogaway P. Optimal asymmetric encryption[C]//Advances in Cryptology—EUROCRYPT'94. Springer Berlin Heidelberg, 1995: 92-111.

[14] Gisin N, Ribordy G, Tittel W, et al. Quantum cryptography [J]. Reviews of modern physics, 2002, 74(1): 145.