An Improved Timing Attack with Error Detection on RSA-CRT*

CHEN Cai-Sen, Wang Tao, Tian Jun-Jian

Department of Computer Engineering, Ordnance Engineering College, Shijiazhuang
caisenchen@163.com

Abstract. Several types of timing attacks have been published, but they are either in theory or hard to be taken into practice. In order to improve the feasibility of attack, this paper proposes an advance timing attack scheme on RSA-CRT with T-test. Similar timing attacks have been presented, such as BB-Attack and Shindler's attack, however none of them applied statistical tool in their methods with such efficiency, and showed the complete recovery in practice by attacking on RSA-CRT. With T-test, we enlarge the 0-1 gap, reduce the neighborhood size and improve the precision of decision. The most contribution of this paper is that our algorithm has an error detection mechanism which can detect the erroneous decision of guessing q_k and correct it. Experiment results show that we could make the success rate of recovering q to be 100% indeed for interprocess timing attack, recovery 1024 bits RSA key completely in practice.

Keywords: Timing attack, RSA, T-test, Montgomery reduction, Chinese Remainder Theorem, Error detection

1 Introduction

Cryptography offers several algorithms that are considered safe on the theoretical level. However, there may be some deceptive signs that expose the algorithms to potential attacks on implementation level. Timing attack is one of the side channel attacks where the attacker could break a cryptosystem by measuring the time differences between specific events. Others include power analysis and attacks based on electromagnetic radiation. Unlike the timing attack, these extended side channel attacks require special equipment and physical access to the machine.

Since the idea of Timing attack was first suggested in 1996 by Paul Kocher[1]. There are several papers that present new, or extend existing theoretical timing attack. In 1998 J.-F. Dhem[2] took Timing attack into practice on Smartcard that stores a private RSA key. Schindler[3] presented timing attacks on implementation of RSA exponentiation that employ the Chinese Remainder Theorem (CRT). There are also some papers which use the results of such theoretical papers to attack some

* Supported by the National Natural Science Foundation of China under Grant No. 60772082; the Natural Science Foundation of Hebei Province under Grant No. 08M010.

algorithms in practice. OpenSSL is a well-known free (open source) crypto library which is often used on Apache Web Servers to provide SSL functions. In 2003 Brumley and Boneh[4] demonstrated that timing attacks can reveal RSA private keys from an OpenSSL-based web server over a local network. In 2005, Onur Aciimez and Shindler[5] proposed an efficient attack on RSA implementations that use CRT with Montgomery's multiplication algorithm, and suggested a general improvement of the decision strategy.

Although there are several implementation of RSA algorithm, such as the well-known left to right square and multiply, CRT algorithm. Here we only focus on the timing attack on RSA-CRT. These timing attack algorithm mostly guess the secret key bit by bit. The advantage of this technique is that it is fast, there are some disadvantages. One is that the attacker couldn't begin guess the bit of q from the first one, Schindler began at the fifth bit, the first few bits are assumed to be determined by BB-attack, in fact they are to be found by using exhaustive search, this will be explained latter, then using the timing attack to guess the remaining bits of q. However, only one of the 31 exhaustive searches makes sense. The other one is that if one single bit is guessed wrongly using timing attack, the following work is meaningless and time consuming, because the next bit to be guessed is based on the hypothesis that the former bits have been guessed correctly, it will not get the correct key. In this paper we propose an advanced algorithm that has an error-detection mechanism. Basically, our algorithm is using statistical data collected by timing measurements to detect the error guess. The experiment result shows that this algorithm could detect the error guess and is an improvement of reducing the number of samples, by a factor of more than 30 for complete recovery key in practice. The most important contribution is, that we improve the feasibility of timing attack on RSA.

The remainder of the paper is structured as follows. Section 2 gives an overview of timing attack, describes the RSA implementation that we are going to use to demonstrate our algorithm, and briefly describes the Chinese Remainder Theorem and Montgomery multiplication algorithm, and explains the statistical tools we are going to use. The general idea of Timing attack on RSA-CRT is introduced in Section 3. Section 4 presents our improved algorithm in details. In Section 5, implementation details are addressed, and the complexity analysis and comparison are discussed between the former attacks and our algorithm. Section 6 is the conclusion.

2 Preliminaries

2.1 Timing Attack

Implementations of cryptographic algorithms often perform computations in non-constant time, due to performance optimizations. If secret parameters are involved in such operations, these timing variations can leak some information and provided enough knowledge of the implementation is at hand, a careful statistical analysis could even lead to the total recovery of these secret parameters. This idea was firstly

proposed by Kocher[1].

Timing attack is one of the Side Channel Attack, it is essentially a way of obtaining some user's private information by carefully measuring the time it takes the user to carry out cryptographic operations. The attacker could measure and use time differences between specific event in the system, once enough information is available, the cryptosystem could be broken. The more accurate the measurements of the attacker are, the errors are smaller, and the fewer time measurements are required.

2.2 Implementation of RSA

2.2.1 RSA

To date, RSA algorithm is still the most popular and secure pubic-key cryptographic system[6]. It is proposed by Rivest, Shamir and Adleman in 1977. Let p and q be two distinct large random primes. The modulus n is the product of these two primes: $n=p*q$. Euler's totient function of n is given by $\phi(n)=(p-1)(q-1)$. And then select a number $1<e<\text{phi}(n)$, where $\text{phi}(n)=(p-1)*(q-1)$, such that $\gcd(e,\text{phi}(n))=1$, and compute d with $d*e=1 \bmod \text{phi}(n)$ using the extended Euclidean algorithm. Here e is the public exponent and d is the private exponent. The encryption is performed by computing $C=M^e \pmod n$, where M is the plaintext and C is the ciphertext from which the plaintext M can be computed using $M=C^d \pmod n$.

2.2.2 Chinese Remainder Theorem

The heart of RSA decryption is a modular exponentiation $M=C^d \pmod n$. OpenSSL uses the Chinese Remainder Theorem to perform this exponentiation. The CRT is described as follows [6].

Algorithm 1 Chinese Remainder Theorem

Input: p, q, d, n and C

1: $C_p = C^{d \bmod (p-1)} \bmod p$, $C_q = C^{d \bmod (q-1)} \bmod q$;

2: $u = q(q^{-1} \bmod p)$, $v = p(p^{-1} \bmod q)$;

3: $M = CRT_{(p,q)\to n}(C_p, C_q) = u.C_p + v.C_q \bmod n$;

4: Return M。

RSA decryption with CRT gives up to a factor of four speedup than normal modular exponentiation, such as left to right square and multiply, making it essential for competitive RSA implementations. RSA-CRT is not vulnerable to Kocher's original timing attack. However, since RSA-CRT uses the factor of N, once the factorization of N is revealed, it is easy to obtain the private key d by computing $d=e^{-1} \bmod \text{phi}(n)$.

2.2.3 Montgomery Multiplication

The sliding windows exponentiation algorithm performs a modular multiplication at

every step. Montgomery Multiplication (MM) is discovered by Peter Montgomery[7] in 1985. It is the most efficient algorithm to compute modular multiplications during a modular exponentiation. It uses additions and divisions by powers of 2, which can be accomplished by shifting the operand to the right, to calculate the result. Since it eliminates time consuming integer divisions, the efficiency of the algorithm is very high.

Montgomery Multiplication is used to calculate

$$Z = abR^{-1} (\mathrm{mod}\, n)$$

Where R is a constant power of 2, $R>n$, and R^{-1} is the inverse of R in module n. A conversion to and from n-residue format is required to use this algorithm. Hence, it is more attractive to use it for repeated multiplications on the same residue, just like modular exponentiations. In order to use Montgomery reduction all variables must first be put into Montgomery form which transforms number x into $xR (\mathrm{mod}\ q)$. So x and y can be multiplied as: $xR*bR=cR^2$. Then, use the fast Montgomery reduction algorithm, to compute $cR*R^{-1}=cR (\mathrm{mod}\ q)$. Note that the result is also in Montgomery form, and thus can be directly used in subsequent Montgomery operations. At the end of the exponentiation algorithm the output is put back into standard form by multiplying it by $R^{-1} \mathrm{mod}\ q$. Montgomery Multiplication Algorithm is shown in Figure 1. The conditional subtraction s-n is called 'extra reduction' [7].

Algorithm 2 Montgomery Multiplication Algorithm

Input: X, Y
$S=\mathrm{MM}(X,Y)=XYR^{-1} (\mathrm{mod}\ n)$

```
1.  S=0;
2.  for i=0 to n-1 do
3.    if X_i is 1 then
4.      S=S+Y
5.    end if
6.    if S is an odd number then
7.      S=S+N
8.    end if
9.    Shift right the binary form of S with one
      position
10. end for
11. if S ≥ n then
12.   S=S-n
13. end if
```

2.3 Statistical tool

In this section, the statistical tools that used in our improved algorithm. The most important tool is the two-sample unpooled T-test. We describe the aim and the main characteristic of T-test. In this paper we use the output values for decision making on the key bits during the data analysis phase of timing attack.

Two-sample unpooled T-test [8] aims to analyze the means of samples from two independent populations. The null hypothesis states that there are no differences between the means of the different groups. The alternative hypothesis is that any of

the group means differ from the others[9]. T-test aims to test the rejection region.

$$H_0 : \mu_1 - \mu_2 = \delta \,, \quad H_1 : \mu_1 - \mu_2 \neq \delta$$

The t-test function is given in Equation 1 as follow.

$$t = \frac{(\bar{x}_1 - \bar{x}_2) - (\mu_1 - \mu_2)}{s_\omega \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}} \tag{1}$$

Where $s_\omega = \frac{(n_1 - 1)s_1^{\,2} + (n_2 - 1)s_2^{\,2}}{n_1 + n_2 - 2}$, $s_\omega = \sqrt{s_\omega^{\,2}}$, $\bar{x}_1$ and $\bar{x}_2$ are the estimated means of the two samples.

Given a confidence level α , it is possible to calculate a two-sided confidence interval with an upper and a lower bound value using the t-test function. If the null hypothesis is true, then the absolute values of these bounds are equal. If the test function output a value within the confidence interval, then the null hypothesis is accepted. If the test function results a value outside the confidence interval, then mean values are considered significantly different. When H_0 is true, the form of rejection region is described as follow.

$\frac{(\bar{x}_1 - \bar{x}_2) - (\mu_1 - \mu_2)}{s_\omega \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}} \geq k$ Where $K = t_{\alpha/2}(n_1 + n_2 - 2)$, rejection region is $|t| \geq t_{\alpha/2}(n_1 + n_2 - 2)$.

3 General idea of Timing Attack on RSA-CRT

As shown in the Algorithm 2, the timing difference is depended whether the extra reduction step is taken. The key relevant fact about Montgomery reduction is at the end of the reduction one checks whether the output *cR* is greater than *q*. If so, one subtracts *q* from the output to ensure that the output cR is in the rang $[0, q)$. The timing difference depended on different inputs. Because Schindler noticed that the probability of an extra reduction during an exponentiation $g^d \pmod n$ is proportional to how close *g* is to *q*[3]. He showed that the probability for an extra reduction is given in Equation 2:

$$\Pr[\textit{extra reduction}] = \frac{g \bmod q}{2R} \tag{2}$$

Consequently, as *g* approaches either factor *p* or *q* from below, the number of extra reductions during the exponentiation algorithm greatly increases. At exact multiples of *p* or *q*, the number of extra reductions drops dramatically. This relationship is showed in Figure 2[3], with the discontinuities appearing at multiples of *p* and *q*. By detecting timing differences that result from extra reductions we can tell how close g is to a multiple of one of factors.

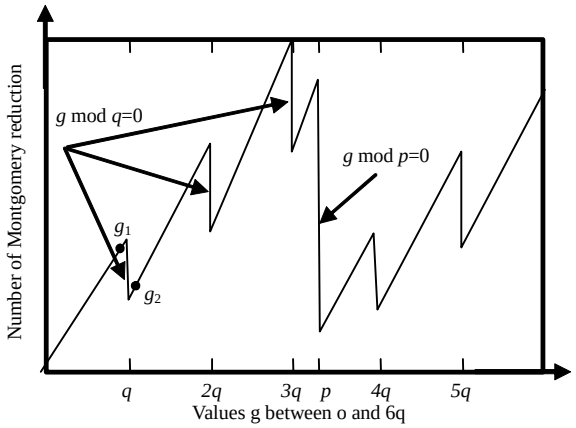

Figure 2 Number of extra reductions in a Montgomery reduction as a function of the input g

As showed in Figure 2, when $g_1<q<g_2$, the difference of the number of Montgomery reduction between g_1 and g_2 is larger than the situation that $g_1<g_2<q$. So we can reduce the search space of q by measuring the timing difference between g_1 and g_2. This is the basic idea of timing attack on RSA-CRT.

BB-Attack[4] exploits the multiplications $MM(temp, \bar{y}_{1,q}; q)$ that are carried out in the exponentiation phase of Sliding Window Exponentiation. Let assume that the attacker has gotten the most significant k bits, he tries to recover q_k, where $q=(q_0,q_1,\ldots,q_{511})$. The attacker generates g and g_i, where $g=(q_0,q_1,\ldots,q_{k-1},0,0,\ldots,0)$ and $g_i=(q_0,q_1,\ldots,q_{k-1},1,0,\ldots,0)$. Note that there are two possibilities for q: $g<q<g_i$ (when $q_k=0$) or $g<g_i<q$ (when $q_k=1$). He decides q_k is 0 or 1 by determining the decryption time $t_1 = T(g) = Time(u_g{}^d \bmod n)$ and $t_2 = T(g_i) = Time(u_{gi}^d \bmod n)$, where $u_g=g*R^{-1}(\bmod n)$ and $u_{gi}=g_i*R^{-1}(\bmod n)$. If q_k is 0, as shown in Figure 2, $g<q<g_i$, then $|t_1-t_2|$ must be "large", otherwise $|t_1-t_2|$ must be "small", which implies that q_k is 1. BB-Attack does not only compare the timings for $gR^{-1}(\bmod n)$ and $g_iR^{-1}(\bmod n)$, but also uses the whole neighborhoods of g and g_i, ie., $N(g,N)=\{g,g+1,\ldots,g+N-1\}$ and $N(g_i,N)=\{g_i,g_i+1,\ldots,g_i+N-1\}$, respectively. The parameter N is called the neighborhood size. We difine T_g and T_{gi} as the time to compute g and g_i with sliding windows when considering a neighborhood of values, calculated as follow.

$$T_g = \sum_{i=0}^{N} DecryptTime((g+i)*R^{-1}(\bmod n))$$

$$T_{g_i} = \sum_{i=0}^{N} DecryptTime((g_i+i)*R^{-1}(\bmod n))$$

The attacker could decide whether q_k is 0 or 1 depending on the relationship of the value of $|T_g - T_{g_i}|$ and a criteria $\triangle$, $\triangle$ is build by the attack. If $|T_g - T_{g_i}| > \triangle$, q_k is 0, else q_k is 1. Repeat the former steps to guess the remaining bits of q, if the upper half of the bit representation of either p or q to factorize n by applying a lattice-based algorithm[9].

More details about BB-Attack can be found in [4]. Whereas Schindler's attack exploits the multiplications with the second power of the base(multiplied with R) in the initialization phase of the table, more detail can be found in [5].

4 Improved Timing Attack with Error Detection

4.1 A new attack model using analysis of T-test with Measurements

This paper we propose a new timing attack model on RSA-CRT, using the statistical tool, such as T-test function, we using the T value instead of $|T_g - T_{g_i}|$ which is used by BB-Attack to decide whether the q_k is 1 or 0. The values generated by the T-test function need to be analyzed before using them in the decision making on the correctness of guess. According to T-test function, when the difference between the means of two groups, the T value will within the confidence interval, otherwise outside the confidence interval. Suppose we already get the top *k*-1 bits of *q*, Let *g* be an integer that has the same top *k*-1 bits as *q* and the remaining bits of *g* are 0, and g_i is the same of *g*, except the *k*'th set to 1, in our new attack model, we recover the *k*'th bit of q as follows:

1. if the k'th bit of q is 1, then $g<g_i<q$, Otherwise, $g<q<g_i$.
2. Compute R and R^{-1}, and build he u_g and u_{gi}, $ug=g*R^{-1} \pmod n$, $u_{gi}=g_i*R^{-1} \pmod n$, This step puts g and g_i into Montgomery form.
3. Measuring the time to decrypt both u_g and u_{gi}, using a neighborhood size N, we will get two samples: $T=[T_1,T_2,...,T_N]$ and $T'=[T_1',T_2',...,T_N']$, where T_i=DecryptTime(u_g), and T_i'= DecryptTime(u_{gi}).
4. Calculate the T value of T and T' using Equation 2. If $g<q<g_i$, the T value will outside the confidence interval. Thus we use the T value as an indicator for the k'th bit of q.

So we also build a criteria $\triangle t$ according to the confidence interval, if $T_k > \triangle t$, we guess q_k is 0, else q_k is 1.

Our attack model has several advantages on the previous ones, using the T value can enlarge the 0-1 gap when q_k is 0, where the 0-1 gap is the time difference $|T_g - T_{g_i}|$. so we can reduce the neighborhood size, improving the accuracy of guess and the efficiency of attack. Using our attack model, we enlarge the 0-1 gap by a factor of about 2 over BB-Attack, reducing the neighborhood size from 400 into the rang [250,300] for key of different length.

The most interesting feature for the new attack model is its error detection mechanism. With this mechanism, we make timing attack on RSA more practical than previous ones. This will be proposed in the next section.

4.2 Algorithm with error detection

There are some important problems about the previous timing attack, it is difficult to take into practice. Those timing attack algorithms mostly guess the secret key bit by bit. The advantage of this technique is that it is fast, there are some disadvantages. One is that the attacker couldn't begin guess the bit of q from the first one, Schindler began at the fifth bit, even more the top bits are supposed to be known. The first few bits are assumed to be determined by BB-Attack. Because when the number of the known bits of q is small, the difference between g and g_i are large, as shown in Figure 2, when the 0-1 gap is small, we can't decide the value of q_k is 0 or 1. Hence, they are to be found by using exhaustive search, then using the timing attack to guess the remaining bits of q. However, only one of the 31 exhaustive searches makes sense. The other problem is that if one single bit is guessed wrongly using timing attack, the following work is meaningless and time consuming, because the next bit to be guessed is based on the hypothesis that the former bits have been guess correctly. Suppose we made an erroneous decision for the value of q_k. Since the bit guessed of q_k is based on the hypothesis that $\{q_0,q_1,\ldots,q_{k-1}\}$ is guessed correctly, we attempt to decide whether q_k is 1 or 0 will thus not make sense, and the criteria we build will both be meaningless. This remains true for the following bits.

The remarkable mechanism of our attack is that it has an error detection mechanism. We find that if the bit of q_{k-1} is guessed wrongly, then one of the two inequations $g<g_i<q$ and $q<g<g_i$ will be always true. As shown in figure 2, the $|T_g - T_{g_i}|$ will always be "small", in addition, we enlarge the 0-1 gap, so the next $|T_g - T_{g_i}|$ will be always "small". Normally, there aren't 10~20 continuous 1 bits in the random private key of OpenSSL's RSA implementation, the error detection is based on this fact. If appearing a number of continuous "small" 0-1gaps during the timing attack, we decide that an erroneous choice has happened. We also propose an error correction policy using the observation of 0-1 gap distributing.

4.3 Error Correction Policy

When using the error correction policy, success rate of recovering q will be improved. It is necessary to find a way to make the attack robust against an as big error proportion as possible. Errors are always detected based on some criterion value. We decided to use a criterion based on the decision criterion for the next bit value after many trials.

We try to find a solution of the first problem of the previous timing attack, suppose we want to guess the top 5 bits of q by timing attack, So we enumerate 31 instance of the top 5 bits of q, but only one instance makes meaning. For every 31 instance, if we find that there are 10~20 continuous "small" 0-1gaps in one instance, we decide this instance is meaningless, stop the attack for this instance and try the next instance. Using this solution we get the first top 5 bits of q, but nor for every instance we try the whole timing attack, it is quite time-consuming.

For the second problem, the solution is similar to the first one. We trace the decision criterion back from the first bit Q_1 which the continuous "small" 0-1gaps

start from. And then alter the criteria to change it's value, then the next bit Q_2's value,…,until there are not a number of continuous "small" 0-1gaps, but not exhaustive search the instances of a window of bits as the first solution.

Basically, the error correction policy is described as following:

```
1. For (2^k-1 instances of the top k bits of q)
     {
2. For every instance, start by performing the attack
   until guessing the 20th bit of q, without any
   correction.
3. Get the first 20 T value, analyze the with the
   decision criterion.
4. if appear 10~20 continuous "small" 0-1gaps
5.  we decide this instance is meaningless, and continue
   next instance.
6. else
       for(i=k; i<the number of guessing bits of q;i++)
      {
   i. continue the timing attack as before, and check
        that whether appearing 10~20 continuous "small"
        0-1gaps.
  ii. If so
      {
      (1) Try to correct the value of the bit Q1 which the
          continuous "small" 0-1gaps start from, and try
          to change the bit value at position Q2, then
          Q3,…, repeat the attack, until there are not
          continuous "small" 0-1gaps.
      (2) If there are still continuous "small" 0-1gaps
          in this way, we conclude the first error
          occurred before Q1, or else q has 10~20
          continuous bits which are all 1, but the
          probability is very small. We thus find the
          last place, before Q1 and change the bit value,
          restart the same process.
      }
      Else continue;
    }
   }
```

The above algorithm will try to detect these errors and correct them. Of course, every time the upper half of the bit representation of q is guessed, we can factorize n by applying a lattice-based algorithm, and then get the private key d using calculate $d=e^{-1}(\mathrm{mod}\ (p\text{-}1)(q\text{-}1))$, check whether it is the right one and stop as soon as the key is found.

5 Experimental Results and Complexity Analysis

5.1 Environment configuration

The same as BB-Attack, we take the OpenSSL as the target of timing attack, but the version of OpenSSL is 0.9.8b [11], All test are run under Linux Fedora on a Intel P4 3.00GHz processor with 1 GB of RAM, using gcc 4.1.2. All RSA keys are generated at random via OpenSSL's key generation routine. The environment configuration of experiment is shown in Table 1.

Table 1 Environment configuration of Timing attack on RSA-CRT

Configuration Item	Parameter
Operating System	Fedora Linux 8
CPU	Intel(R) Pentium(R) 4 CPU 3.00GHz
Gcc	gcc version 4.1.2
Cryptographic Library	OpenSSL v0.9.8b

5.2 Experimental Results

Under the environment configuration shown in Table 1, we performed a series of experiments to demonstrate the effectives of our attack on OpenSSL, especially the error detection mechanism of our attack. In order to get the time of decryption accurately, we use the Pentium cycle counter on the attack machine as such a clock, giving a time resolution of 3.0 billion ticks per second. Thus, the decryption time is the cycle counter which is accessible via the "rdtsc" instruction [12], which returns the 64-bit cycle count since the CPU initialization.

Firstly, we repeat BB-Attack, suppose that the top 5 bits of q has been know, but we using the T-value which has been introduced in section 2.3, instead of $|T_g - T_{g_i}|$. Using the T-value to decide the bit of q_k is 0 or 1 by a decision criterion. During the experiment, we use a sample size of 7 and a neighborhood size of 300, resulting in 1075200 total queries for BB-Attack on 1024 RSA bits. With these parameters a typical attack approximately 65 minutes in the abstract. We could recover all the bits of q by timing attack, but it is not necessary, if we get the upper half of the bit representation of q, n could be factorized by applying a lattice-based algorithm [9]. As we suppose that the top 5 bits of q has been know, the distribution of T-value of bits guessed of factor q from 6^{th} to 106^{th} is shown in Figure 3.

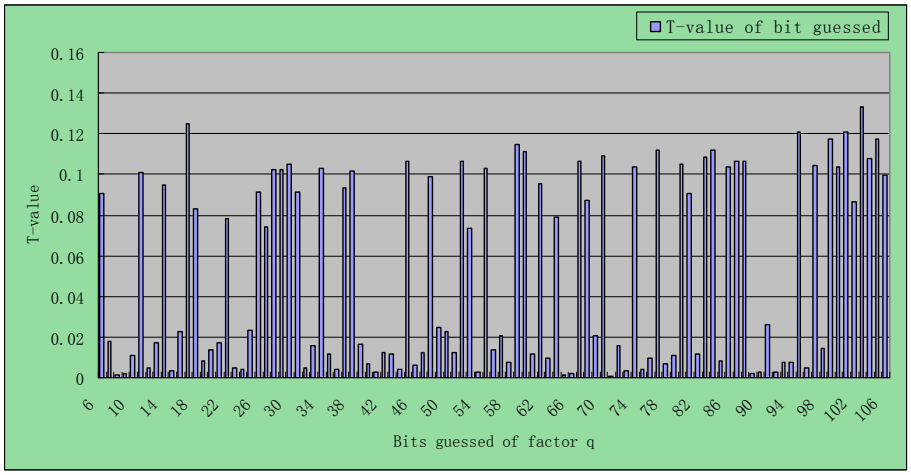

Figure 3 The distribution of T-value of bits guessed of factor q from 6th to 106th

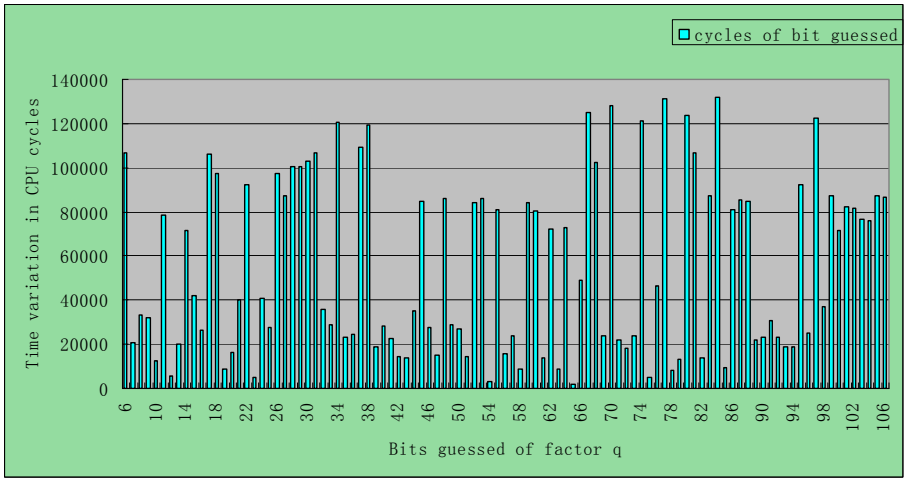

Figure 4 The distribution of $|T_g - T_{g_i}|$ of bits guessed of factor q from 6th to 106th

As shown in Figure 3, Using the decision criterion of T-value 0.04, we can recover the top 106 bits of q completely. And the 0-1 gap is about 200% larger in our attack.

In practice, during the timing attack, we performed interprocess attacks, the precision of time decryption may be affected by the noise caused by other process in operation system or the confliction of software and hardware. Clearly, in network timing attacks the noise may be much larger, and hence an attack may become impractical even if it is feasible for an interprocess attack, under the same environmental conditions. The noise may result in an erroneous decision for the value of bit q_k, affect BB-Attack and Shindler'attack as well. But the previous timing attacks without error correction policy, it is hard to take them in practice. We propose an algorithm for error detection, the result is shown in Figure 5, when the 98th bit of q is guessed wrongly, it is obviously that there are more than 10 continuous "small" 0-

1gaps after 98th, we can conclude that the 98th guessed bit may be wrong, so then we try the error correction policy to correct the erroneous decision. We can try to alter the criteria to be 0.03, so the 98th bit of q can be guessed as 0 criteria, and continue the attack of 99th bit of q.

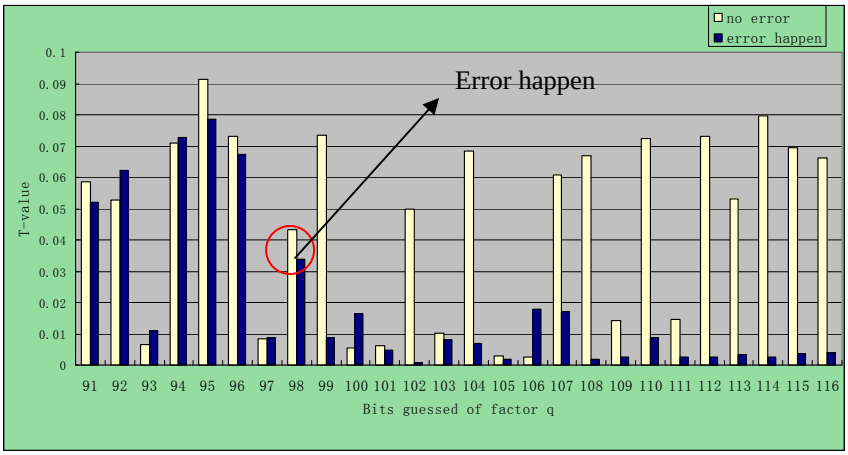

Figure 5 Detection of an error at the 98th bit guessed of q for a 1024-bit RSA key

There are two things worth about the experiment: Firstly, it can be seen that when we use T-value instead of $|T_g - T_{g_i}|$, it can enlarge the 0-1 gaps, making the 0-1 gaps are larger than before, so 0-1 gaps are easy to be distinguished, we improve the precision of decision. Secondly, thanks to the error detection mechanism, we can build an attack model that would be robust against errors and correct them, improving the practice of timing attack.

5.3 Complexity Analysis and Comparisons

We focused on improvements able to improve success rate of recovering a secret key by timing attack. The previous timing attacks are based the on the hypothesis that the top 5 bits of q have been know, but how to get the top 5 bits? In fact they are to be found by using exhaustive search, then using the timing attack to guess the remaining bits of q. If using the error correction policy, it could reduce the sample size for successfully by a factor about 31 in theory.

The performance of timing attack are highly environment dependent, therefore it is not reliable to compare the figures of two different attacks running on different systems[5]. When we perform the BB-Attack and Shindler'attack in the same environment shown in Table 1, under the precondition that the top 5 bits of q have been know, our attack aims to get the bits of q from 6th to 256th, and calculate the success rate. For interprocess attacks, the probability for correct guesses is in the rang [91%,96%], and the error guesses mostly happen in the top bit from 6th to 32th when we attack the 1024 bits RSA key in OpenSSL. In the attack of these bits, we must increase the neighborhood size to reduce the difference between g and g_i, the neighborhood is about 500, while it is about 300 for the bits from 100th to 256th. For network attacks, the probability for correct guesses is lower than above, is in the rang

[81%,90%], and even lower, with noise caused by network delay times additionally. The success rate of recovering a secret 1024 bits key is shown in Table 2.

Table 2 Success rate of recovering q

Timing Attack	Success Rate			
	Interprocess attack		Network attack	
	Without error correction	With error correction	Without error correction	With error correction
BB-Attack	91%~96%	99%-100%	81%~90%	90%-95%
Shindler'attack	93%~97%	99%-100%	82%~90%	91%-95%

The Shindler'attack can give an improvement by a factor of more than 5 over BB-Attack[5], but it could not improve the success rate of attack so much. For network attack, as the noise caused by network delay is much large, affecting the time precision badly. With error correction, we make the success rate of recovering q to be 100% indeed, whereas more samples will be need in practice.

6 Conclusion

We have proposed an advanced timing attack on RSA-CRT with error correction policy. Our experimental result shows that using T-test statistical tool, it could enlarge the 0-1 gap to reduce the neighborhood size and improve the precision of decision; most important is that our attack algorithm has an error detection mechanism. This mechanism has been demonstrated in the experiment of timing attack on RSA-CRT of OpenSSL in practice. With error correction, the 1024bits RSA key can be completely recovered in interprocess timing attack in practice.

References

1. Paul Kocher. Timing attack on Implementations of Diffie-Hellman, RSA, DSS, and Other systems[A]. Proceedings of Advances in Cryptology-CRYPTO'96, Springer-verlag, 1996:104-113.
2. J.-F.Dhem, K.Koeune, P.-A. Leroux, P. Mestre,J.-J.Quisquater and J.-L. Williams. A practical implementation of the timing attack. In proceedings of CARDIS 98, or University Catholique de Louvain, Crypto Group Technical Report.
3 W. Schindler. A timing attack against RSA with the Chinese remainder theorem,Proc. of Cryptographic Hardware and Embedded Systems (CHES 2000) (C. Paar, eds.), Springer, 2000, LNCS 1965: 109-124.
4. Brumley.D and Boneh.D. Remote timing attacks are practical[J],Proceedings of the 12th Usenix Security Symposium, 2003,4(12):1-14
5. Onur Acıiç¸mez,Werner Schindler. Improving Brumley and Boneh Timing Attack on Unprotected SSL Implementations Oregon State University Corvallis[A], USA. CCS'05, November 7–11, 2005, Alexandria, Virginia, USA.Copyright 2005 ACM 1595932267/ 05/0011
6. A.J. Menezes, P.C. van Oorschot, S.C. Vanstone:Handbook of Applied Cryptography, Boca Raton, CRC Press 1997.
7. Montgomery.Peter . Modular multiplication without trial division[J]. Mathematics of Computation, 1985,44(170):519–521

8. J. M. Utts and R. F. Heckard, Statistical Ideas and Methods. South Melbourne, Victoria 3205, 80 Dorcas Street, Australia: Cengage LearningAustralia, 2006.Rudolf Toth,Zoltan
9. Faigl,Mate Szalay,Sandor Imre. An advanced Timing attack scheme on RSA.In Proceedings of the 13th International Telecommunications Network Strategy and Planning Symposium,NETWORKS 2008.Budapest,Magyarorszag,2008.09.18-2008.10.02 pp.1-24.(ISBN:978-963-8111-68-5).
10. D. Coppersmith: Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities. J.Cryptology 10 (no. 4) (1997) 233-260.
11. OpenSSL Project: OpenSSL:http://www.openssl.org.
12. Intel. Using the RDTSC instruction for performance monitoring [R]. Amerrica:Intel, 1997.