Journal of Computers Vol. 29 No. 2, 2018, pp. 45-54
doi:10.3966/199115992018042902005

Improved Fault Analysis on RSA Based on Non-invasive Fault

Cai-sen Chen[1*], Yang-xia Xiang[2], Zhi-wei Cheng[2], Jia-xing Du[2], Xiang-liang Ma[3]

[1] Department of Training Center, Academy of Army Armored Force, Beijing 100072, China
caisenchen@163.com

[2] Department of Information Engineering, Academy of Army Armored Force, Beijing 100072, China
elephant_187@163.com, cheng.zw@mail.scut.edu.cn,dhtl2004@163.com

[3] Institute of Software, Chinese Academy of Sciences, Beijing 100049, China
maxiangliang@163.com

Received 24 July 2017; Revised 19 September 2017; Accepted 19 October 2017

Abstract. First, by analyzing the previous fault analysis on RSA using square and multiply algorithm because these attacks require quite precise fault injections like a bit flip and the fault injection can't be used effectively, this paper builds a new fault analysis model on RSA using square and multiply algorithm based on non-invasive fault, including the square operation fault and the multiply operation fault; Next, an improved fault analysis algorithm based on recovering key segment and a random fault analysis algorithm on RSA using SPA-FA resistant are proposed separately, these proposed fault analysis algorithms are proved in theory and the method of controlling the opportunity in fault injection is suggested; Finally, a fault attack is implemented on RSA in a Protues simulation environment, using the computer program simulation to inject the fault. Experimental results show that the proposed fault analysis algorithm based on recovering key segment can reduce the accuracy requirements of fault injection, and number of fault injection is reduced by about 80%; and the random fault analysis algorithm based on the multiply operation fault can cancel the dependence between the each key bit recover, further improve the feasibility of fault analysis.

Keywords: fault analysis, non-invasive fault, public-key cipher, RSA, side channel attack, square-and-multiply algorithm

1 Introduction

The cipher algorithm is often applied to security of cipher algorithm. There are side channel attacks and security holes of fault analysis in cipher algorithm, and some fault analyses of RSA algorithm which focus on vulnerability about the Chinese Remainder Theorem (CRT); the others are implemented by non-assisted methods. For example, square-and-multiply is used for modular exponentiation algorithm. However, the latter need to inject fault accurately. For example, fault is made of one bit or fault is injected to special operation.

Execution attacks focus on execution of cipher algorithm instead of algorithm and are different from traditional algorithm, which regard algorithm as gray-box. Although cipher algorithm is secured from mathematical analysis, executions about algorithm and device may leak much execution information and attacker can get inner state and execution flow by side-channel. In 1997, fault analysis was proposed firstly by Boneh, which has serious threat to cipher device [1]. RSA key are restored completely by fault which is made by cipher algorithm. And then this attack idea was improved by Biham and Shamir, which can restore key with clear-text and incorrect signature [2]. Fault analysis of square-and-multiply on RSA is proposed by Bao [3], in which key leak is caused by exponential fault during modular exponentiation

* Corresponding Author

operation. In 2005, fault analysis on RSA with right-to-left square and multiply algorithm was put forward by Serfert [4], which is inject fault for module(N) during model-power operation. In 2009, fault analysis on RSA with left-to-right square-and-multiply algorithm was put forward by Berzati [5], and fault analysis on RSA with square-and-multiply was showed by Schmidt [6], which was implemented by transient clock inference. In 2009's FDTC, RSA attack model on ARM9 was proposed by Barenghi [7], which injected fault by voltage. In 2009's RSA, RSA on FPGA was put forward by Pellegrini [8], which was implemented by changing power voltage.

In this paper, based on previous studies, security on RSA fault analysis is implemented by square and multiply algorithm, weakness of previous attack algorithm are analyzed, fault analysis model is built with executing fault using skipping some instructions, precision of injected fault is much lower, key analysis algorithm on injected fault is used, and results are verified by simulations. Experiment results show that fault is injected to one attack key segment instead of special key bit, random fault analysis model are gave with injecting fault in multiply operation, which not only improve efficiency of fault analysis, but also can be attacked successfully RSA algorithm with SPA-FA.

The contributions of this paper are followed as:

(1) On basis of previous attack algorithm, a new fault analysis algorithm is proposed, which need much lower precision of injected fault and is implemented easily. And single chip program counter can make mistakes that inner reason and possibility is analyzed, and it provides method for attack. The new algorithm uses key segment as analysis object instead of key bit, and improves possibility for injected fault operation.

(2) Random fault analysis algorithm on square operation is proposed, which is implemented with executing fault using skipping some instructions. Usability of injected fault is improved; times of injected fault are reduced, attack cost is lessened, and threat and reliability is enhanced. The algorithm is applied to RSA with SPA-FA and hardware, complexity of attack is analyzed in detail, and results are verified by simulations.

The structure of this paper is as follows: RSA algorithm and relative knowledge of fault attack is introduced in section 2; fault analysis model and idea on RSA are given in section 3; the improved fault analysis algorithm is proposed and innovation is mainly described in section 4; complexity of attack and results are analyzed by simulations in section 5; Conclusion in section 6.

2 Related Work

2.1 RSA Algorithm

RSA is a kind of public key cryptography algorithm, which uses public key (N,e) for encrypting and private key (d) for decrypting. Modular (N) is equal to $p*q$, where p and q are both primes. The equation is $e*d$=1mod $(p-1)(q-1)$, where e and d are both exponents. The public key on RSA is comprised of (N,e) and d. The cryptography equation is $C=M^e \bmod N$, decryption operation is $M=C^d \bmod N$, where d is private key for decrypting and C is cipher text. To speed up model exponentiation, CRT is used, which is 4 times faster than others. In this paper, single chip and AVR is used, and RSA can be implemented easily, square and multiply algorithm is applied to modular exponentiation operation. There are two kinds of square-and-multiply algorithm: Left-to-Right and Right-to-Left. See Fig. 1 and Fig. 2.

Algorithm 1. "Left-to-Right" square-and-multiply algorithm

Input: m, N, $d=(d_{n-1}, d_{n-2}, \cdots, d_1, d_0)$
Output: $S = m^d \bmod N$

```
(1)S=1
(2)For k from n-1 to 0 do:
   2.1  S = S*S mod N
   2.2  if dk==1, then S=S*m mod N
(3)Return S.
```

Fig. 1. Left-to-Right square-and-multiply algorithm

Algorithm 2. Right-to-Left square-and-multiply algorithm

Input: m, N, d=(d_{n-1}, d_{n-2}, ⋯, d_1, d_0)
Output: S = m^d mod N

(1) S=1
(2) For k from 0 to n-1 do:
  2.1 if d_k==1, then S=S*m mod N
  2.2 m = m*m mod N
(3) Return S.

Fig. 2. Right-to-Left square-and-multiply algorithm

2.2 Fault Attack

Fault attack is one kind of attack that device is not work by attacker and calculation is fail, and is of side-channel attack [1]. For RFID, fault attacks make one or many triggers failure and destroy data in register and memory. Fault attacks are applied to IC widely. The attack procedure is consisted of fault induction and fault analysis.

Fault induction injects fault to one location of device at proper opportunity, Work environment and device have impact on technology and effect. Requirements for fault induction include: voltage, clock, temperature, radiation, hard light, vortex and so on.

All or parts of key are recovered by special analysis method using wrong results, which is called fault analysis. In most cases, fault analysis depends on cipher system design and implementation, cipher algorithm rules and induced fault types; and fault analysis combines with traditional cipher analysis method. Among fault analysis on public key algorithm, the main analysis methods are greatest common divisor and collision analysis, object of study is modular exponentiation operation. In this paper, RSA fault analysis on square-and-multiply algorithm is analyzed.

For attack methods, there are many kinds of fault analysis: intrusive attack, half-intrusive attack and non-intrusive attack. The first two are used storage data instead of the transmission procedure fault, so direct attack objective is applied, exfoliation device and chemical substances are expensive, and FIB is applied to special device for fault injection. Using extern inference, non-intrusive attack make attack objective failure, which are transient changing voltage and magnetic field interference and so on. This method has much feasibility.

2.3 Non-intrusive Fault Injection

Non-intrusive fault injections have an effect on trigger and have little effect on machine cycle, which are transient clock, transient electricity and transient extern magnetic.

Clock fault is done by increasing and reducing one or half-cycle of clock rate. For example, many impulse peaks are injected to CPU clock, and inputs are sampled by some triggers before new states are coming, data transmission is fail between register and storage [9]. Clock fault attack combines with electricity fault, and PC is increased using clock and mains fluctuation in touch IC. Instruction sequence of IC can be executed by attacker.

Components, which are ALU, PC, IR, register file and FR, are sensitive to AVR. Research shows that ALU, PC, IR are the most sensitive. ALU is the primary component and is susceptible to fault [10].

In this paper, PC is interference with clock rate or voltage, and PC is fail. So some instructions are missed or decoding is invalid.

2.4 Attack Hypothesis

Attack hypothesis is as follows:

(1) Different cipher algorithms are done by different analysis algorithm, so the hypothesis is that those algorithms on each cipher algorithm are known by attacker.

(2) Fault is injected by missing some right instructions, so the hypothesis is that fault location and size are controlled by attacker.

(3) Characters of injected fault are known by attacker, those are not identified by correct signature S and incorrect signature S'. If the injected fault is the special fault, the relative key bit or key segment are recovered. Attack objective do deciphering by attacker and cipher text are gathered.

3 Fault Analysis Model

3.1 Fault Injection

Based on fault injection using transient clock, Schmidt executes modular exponentiation of RSA with square-and-multiply algorithm, makes program counter failure in operation process by ignoring one square operation, and gets wrong results [6]. Considering RSA with SPA-FA and missing multiply operations, attack model is shown Fig. 3.

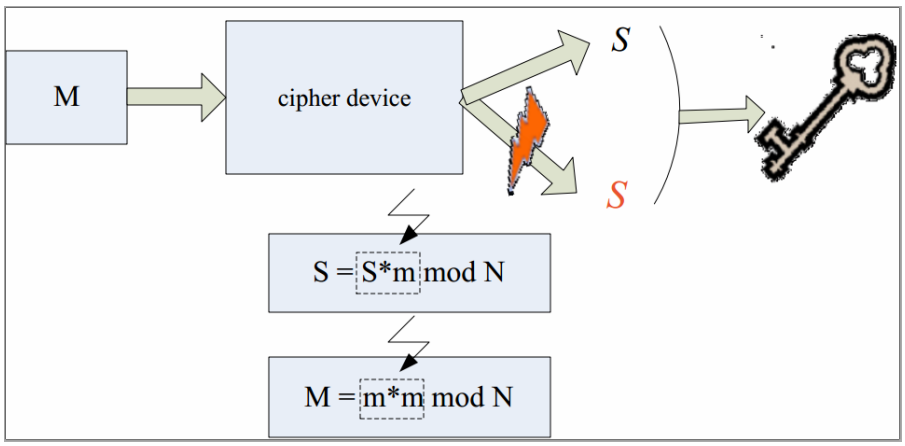

Fig. 3. RSA fault analysis model with square-and-multiply algorithm

In Fig. 3, dashed box represents operation step that may overlook. The time of fault inject is the i^{th} loop operation of square-and-multiply algorithm, fault location is square or multiply operation.

3.2 Attack Basic Idea

The main attack idea is ignoring square operation on RSA algorithm with injecting fault. Based on "divide and rule" strategy, key-restoring algorithm recovers the value of key by bit or by segment using effectively injected fault result, and completed keys are restored by repeating execution. To describe attack algorithm, Left-to-Right is took as a case study, the algorithm can be applied to other algorithm. For example, RSA is implemented by Right-to-Left square-and-multiply modular exponentiation or sliding window algorithm. Whatever multiplies operation algorithms are executed by square-and-multiply or square-and-sum, one square operation is missed in attack, which is able to change result. Results are depending on modular exponentiation, every fault injection will result in leaking with one key bit or key segment.

4 Improved Fault Analysis Algorithm

4.1 Related Research

In 1997, RSA fault analysis was implemented by CRT algorithm, which was proposed firstly by Boneh [1]. And fault analysis is focused by cryptanalyst. In this paper, research on fault analysis is done by RSA square-and-multiply. Invasive attack is fault injection by modifying modulus N [4-5], non-invasive are made multiplier failure by voltage [8] and made program missing instruction execution by transient clock [6]. In 2000, security error attack was proposed by Yen and Joye [12]. Using fault injection precisely in register, responsive power exponent value was deduced by observing result, whether result is correct or not.

However, many attacks are based on theory model, and the feasibility is not strong. The first several kinds of attack need bit fault precisely, and the last one attack special operation, whether fault is injected or not. In 2006, one kind of attack algorithm was proposed by Boreale, which was between fault injections precision time and square-and-multiply median using one kind of random fault that is controlled time, and square result was replaced by random value in one register [11]. If there is 5000 fault samples, 70% key are recovered, which is 1024 bit.

In FDTC 2008, fault analysis algorithm on square-and-multiply was proposed by Schmidt [6]. In this paper, the improved algorithm is put forward basing on the above algorithm. Steps are as follows:

The correct signature is $Sig=m^e$. Assumed that Sig_k, if $e_i=1, k \in \{0,...,t\}$ represents fault signature that miss the $(t\text{-}k+1)^{th}$ square operation. If the initial values of intermediate variable is 1 and the first square operation is missed, which are affect outcomes, so $Sig_t=Sig$. If other fault signature is Sig_k, $k \in \{0,...,t-1\}$ is as follows:

$$Sig_k = \prod_{i=k+1}^{n} m^{e_i 2^{i-1}} * \prod_{i=0}^{k} m^{e_i 2^i} \bmod n \tag{1}$$

Key bit is recovered from the right-most bit, which is start with fault signature (Sig_0). The correct signature (Sig) is given by Sig_0 and e_0. If fault can control correctly using above attack model, relationship is as follows:

$$Sig = \begin{cases} (Sig_0)^2 \bmod n & for\ e_0 = 0 \\ (Sig_0)^2 * m^{-1} \bmod n & for\ e_0 = 1 \end{cases} \tag{2}$$

If fault is injected again, e_0 can be got. And then the next fault signature is deduced and the value of power exponent is got. In both cases, relationship between fault signatures is got:

$$Sig_k = \begin{cases} Sig_{k-1} & for\ e_k = 0 \\ Sig_{k-1} * m^{2^{k-1}} \bmod n & for\ e_k = 1 \end{cases} \tag{3}$$

e_k can be deduced by relationship of adjacent fault signature. If equation is not match, fault signature is failure, otherwise e_k can be got and complete key can be recovered by above execution repeatedly.

If e_k is 0, the present fault signature is equal to the prior one. If the same square operation is missed many times, the key bit is 0 that is wrong. If key bit have a series of 0, attacker may not get accurate location. The disadvantage of algorithm is needed to inject fault for special operation continuous. To reduce the demands for fault injection, one kind of algorithm is proposed in this paper, which is based on square operation fault and multiply operation random faults.

4.2 Fault Analysis Algorithm Based on Square Operation Fault

Compare with the previous algorithms, a new method of fault injection is given in this paper, which deduce key on RSA by key segment instead of bit by bit. And attacker can recover key bit by injecting fault bit at random location. Steps are as follows:

The RSA algorithm is analyzed, which is implemented with Left-to-Right square-and-multiply algorithm. $d=(d_{n-1},d_{n-2},\ldots,d_1,d_0)$, so $S=\prod_{i=0}^{n-1} m^{d_i 2^i}$, and d was divided into two segments, one is $d_{(1)}$, the other is $d_{(2)}$, where $d_{(1)}$ is high-order n_1 bits, and $d_{(2)}$ is low-order$(n\text{-}n_1)$ bits. Suppose that the highest-order bit of $d_{(2)}$ is d_k, then $S = m^{d_{(1)} * 2^{k+1} + d_{(2)}}$. When the square operation of d_k is fail, that if d_k is 0, $S_k = m^{d_{(1)} * 2^k + d_{(2)}}$; And when d_k is 1, $S_k = m^{d_{(1)} * 2^k + d_{(2)} - 2^k}$. So, the relationship between the fault signature and correct signature is as follow:

(1) When square operation of dk is fail, the relationship between fault signature S_k and correct signature S is $S = S_k * m^{d_{(1)} * 2^k}$, which is represented as shown below:

$$S = S_k * m^{(d_{n-1},d_{n-2},...,d_{k+1}) * 2^k} \tag{4}$$

(2) Similarly, the relationship between S_{k-1} and S is as follows:

$$S = S_{k-1} * m^{(d_{n-1},d_{n-2},\ldots,d_{k+1},d_k)*2^{k-1}} \quad \textbf{(5)}$$

(3) When d_k=0 and S_k=S_{k-1} ，the relationship can be stated as follows:

$$S = S_k * m^{(d_{n-1},d_{n-2},\ldots,d_{k+1},d_k)*2^{k-1}} \quad \textbf{(6)}$$

In this way the attacker can get the value of $d_{(1)}$ by exhausted search, $d_{(1)}$ and d_k can be calculated by equation (6), and the complete key d can be deduced. The above algorithm can be applied to RSA algorithm implemented with Right-to-Left square-and-multiply algorithm.

4.3 Random Fault Analysis Algorithm Based on Multiply Operation Faults

If a transient clock is changed, a square-and-multiply operation can be ignored. We analyzed the SPA-FA algorithm, which was proposed by Boscher [13]. The multiply operation is executed in every loop, by using the correlation of temporary variable and the existing relationship is verified, to avoid the fault output result. The algorithm is described as follows:

Algorithm 3. Right-to-Left modular exponentiation algorithm on anti-SPA-FA

Input：M, d=$(d_{n-1},d_{n-2},\ldots,d_0)_2$,$N$ Output：M^d mod N or “Error”
1. S[0]=1 2. S[1]=1 3. A=M 4. for i from 0 to n-1 do 5. { 6. $S[\overline{d_i}] = S[\overline{d_i}].A \bmod N$ 7. A=A^2 mod N 8. } 9. if (M.S[0].S[1]=A mod N) and ($A \neq 0$) then 10. return S[0] 11.else 12. return Error

From the algorithm 3, whether the bit d_i is 1 or 0, the multiply operation is always executed, so it can resist Simple Power Analysis. Furthermore, as in every loop calculate, the equation $M.S[0].S[1] = A \bmod N$ is still contented, so we can check this equation to judge whether a fault is happened, to resist the fault analysis.

The step 4 to step 7 in algorithm 3 are repeated as fellow:

(1) When d_i=0, calculations are executed as follows:

$S[1] = S[1].A \bmod N$ and $A = A^2 \bmod N$.

(2) When d_i=1, calculations are executed as follows:

$S[0] = S[0].A \bmod N$ and $A = A^2 \bmod N$.

So if a multiply operation $S[\overline{d_i}] = S[\overline{d_i}].A \bmod N$ corresponding with d_i is ignored, when d_k=1, the relationship between fault signature (S'_k) and correct signature is denoted as follow:

$$S = S'_k * m^{2^k} \quad \textbf{(7)}$$

If d_k=0, the injected fault has no effect on the output results. Based on these outputs, the attacker can judge d_k whether is 1 or 0. The analysis model can inject random fault, it has no dependency on different key bits. Fault signature of key bit is stored in queue, which is used for the next fault signature.

Based on the relationship between the fault signature and the correct signature is shown as equation (8), if d_k is given recurrently, the incorrect signature which is got with square-and-multiply operation that can be ignored.

$$S'_k = S_k * m^{(d_{n-1}, d_{n-2}, \ldots, d_{k+1}) * 2^k - 2^k} \quad \textbf{(8)}$$

Based on equation (8), the incorrect signature of square operation fault can be checked with multiply operation random fault model, and square operation fault algorithm can be done without adjacent fault signature. The incorrect output is got by square operation fault algorithm, which can check the incorrect signature of multiply operation faults. Combined with these above two algorithms, utilization of injected fault can be improved, and feasibility can be also advanced.

4.4 Time of Fault Injection

In this paper, the improved analysis algorithm needs not to inject fault for d_k, but the fault location need to be inject within the key segment d. Consider that the time for every modular exponentiation operations is not equal, multiply-and-square operations are defined as μ_j and v_j separately and the time for instruction control flow is ignored. The time difference for the same operations can be related to instruction queue, random delay or blind parameter and so on. Simplifying these random, device execution time is obscure to attacker. T_i is as follows:

$$T_i = \sum_{j=0}^{i-1} (d_j . \mu_j + v_j) + 1, \ 0 \le j \le i-1 \quad \textbf{(9)}$$

Where $0 \le j \le i-1$, d_j's denotes value of key segment, μ_j's and v_j's are continuous random variant. With reference [15], variants are normal distributions, and the variance and the mean are given. All random variants (μ_j, v_j,) are independent in pairs and are irrelevant to d_i. Time for fault injection (T) is $T_i > T > T_i - v_{i-1}$, the middle time of square operation in $(i\text{-}1)^{th}$ step is as follows:

$$t = T_{i-1} + d_{i-1}\mu_{i-1} + v_{i-1}/2 \quad \textbf{(10)}$$

So assume that the loop operation time for d_k is equal, the time for multiply operations is analyzed by device power consumption and timing that is proposed by Aumuller [14]. Modular multiplication for hardware has more power consumption than ALU, so interval for modular exponentiation is got by attacker. Step 3 to step 6 in algorithm 3 consume the most time in all operations, so the time of fault injection can be controlled easily.

5 Experiment Results and Analysis

5.1 Experiment Results

There are two phrases in experiment. The first phrase is used to verify the correction of attack algorithm based on injected fault using software. The second one is used to verify the feasibility of attack algorithm, Proteus simulates 89C52 and fault is injected by sudden clock rate change.

In the first phrase, RSA on square-and-multiply modular exponentiation is done by VC++ 6.0 and cipher library of OpenSSL on PC (CPU is AMD Athlon 64 3000+, memory is 1G). Fault is injected by software, so the size, location and time of fault injection can be controlled.

When the sample sizes, fault locations and segment sizes are set up, and fault for square and multiply operations could be executed. In our experiments, if every fault is injected correctly, which is injected from the first loop to the last one, the corresponding key bit can be recovered by injected fault. For a 1024-bit key on RSA, the head-end bit for key are both 1, so complete key can be completely recovered by 1022 times for fault injection. To know the time of fault injection, the time of signature by key using OpenSSL and the time for multiply operations in square-and-multiply algorithm are given in Table 1. If the length of key bit is much longer, the more times execution is need. Modular is much larger, the more time for multiply operation will be. The result for 1024-bit RSA fault is shown in Table 2. Fault injection is reduced by 80% in the above algorithm. There are two kinds of attack models: square operations and multiply operations.

Table 1. Execution time range for different key length

the length of key bit	the range of execution time/(CPU Cycle)	the operation time/(CPU Cycle)
512	9450000~9500000	2400~2600
1024	56500000~58000000	2700~4300

Table 2. Results for RSA fault analysis

Attack Model	Fault Type	Fault Location	Segment Size	NFI	Key Space	Time
Reference [6]	Special fault	Square Operation	1	1022	1022	≈45s
Section 4.2	Random Fault	Square Operation	5	205	32*205	≈96s
Section 4.3	Random Fault	Multiply Operation	1	1022	1022	≈30s

Based on the above experiment results, RSA program on AT89C52 is implemented by Proteus, shown as in Fig. 5.

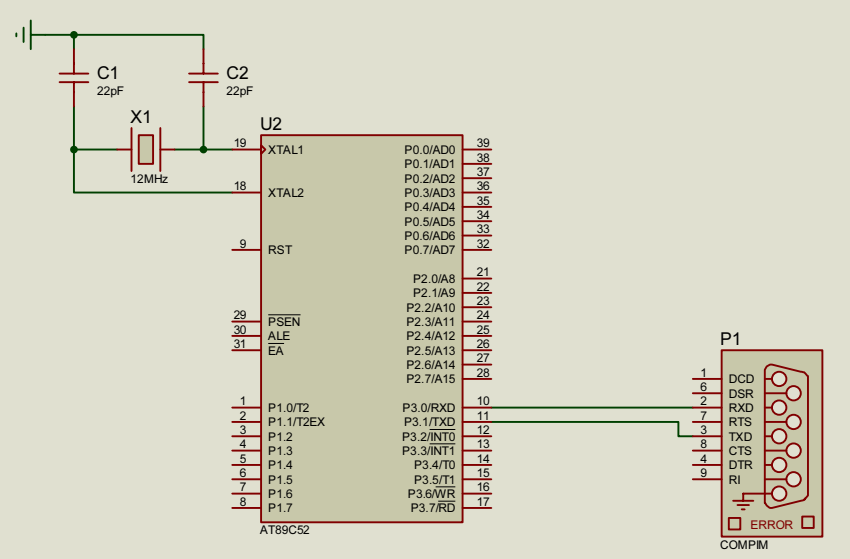

Fig. 5. Fault analysis experiments design drawing on Proteus

Serial communication baud rate is set as 9600Bps, the clock rate of AT89C52 is set as 11.0592MHz, and the timer method is set in mode 2. The highest crystal frequency is 12MHz, which also can be adjustable. As the storage space of single chip is limited, the key length of RSA is only 8-bit in our experiment. Fault location and time is controlled roughly by method in section 4.4, so we use the random fault model to analysis. In experiment, if the clock rate is 11.6412MHz, encryption can work. However, if clock rate is greater than 11.6412MHz, encryption is fail. And if clock rate is less than 10.6200MHz, encryption is unstable. So the extern crystal clock rate of single chip is modified, which can make the RSA encryption fail. 8-bit RSA key space is down from 2^8 to 2^3. The experiment results show that the RSA fault analysis in cipher device by fault injection using changing clock rate is feasible, and the feasibility of fault analysis can be improved as the random fault model.

5.2 Complexity Analysis of Attack Algorithm

Based on the square operation fault model, if the key length of RSA is n and fault is the special one, the number of fault injection is at least (n-2). If the recovery units is key segment and length of segment is x, the number of fault injection is at least n/x. If search space of key segment is 2^x, the complete key is need $2^x*(n/x)$ search space. According to equation (6), the last bit is 0, which is used to recover the key segment. Key segment is random, so complexity of algorithm is also random.

Based on the multiply operation fault model, as only the corresponding key bit can be recovered for every successful fault injection, so the number of fault injection is at least (n-2). The key bit is not dependent on the other fault signature, so the result for fault signature is used effectively.

6 Conclusion

Based on the previous RSA fault analysis, signature fault is occurred, and cipher device is inference by changing the clock rate and some instructions are ignored during the square-and-multiply operations. Because the square operation fault attack has some shortcomings, algorithm on key segment random fault analysis is proposed. For RSA algorithm implemented with SPA-FA, algorithm on multiply operation faults is proposed. Theoretical deduction and simulation lab are both done, and algorithm is verified by OpenSSL. The complete key is recovered successfully by software. RSA fault analysis experiments on AT89C52 are done with Proteus, which can verify the feasibility of attack algorithm. With experimental results and theory of algorithm, complexity of attack algorithm is given. The experimental results are shown that injected fault is lower accuracy with the improved fault analysis algorithm, and the new attack algorithm has more feasibility.

In this paper, attack experiments are implemented by Proteus software. As the practical experiments of fault analysis are much less, so our experiments have more theoretical value and reference function. In the further work, the practical physical experiments will be done by simulated experimental result. And other implementation methods on RSA will be studied further and cipher default detection and protection will be discussed.

Acknowledgements

The authors would like to thank anonymous reviewers for their valuable comments. This research was supported by the National Natural Science Foundation of China under Grant No. 61402528.

Reference

[1] D. Boneh, R. DeMillo, R. Lipton, On the importance of checking cryptographic protocols for faults, in: Pro. International Conference on Theory and Application of Cryptographic Techniques, 1997.

[2] A.K. Lenstra, Memo on RSA signature generation in the presence of faults, manuscript, 1996.

[3] F. Bao, R. H. Deng, Y. Han, A. B. Jeng, A. D. Narasimhalu, T.H. Ngair, Breaking public key cryptosystems on tamper resistant devices in the presence of transient faults, in: Proc. the 5th International Workshop on Security Protocols, 1998.

[4] S.J. Pierre, On authenticated computing and RSA-based authentication, in: Proc. ACM Conference on Computer and Communications Security, 2005.

[5] A. Berzati, C. Canovas, J.G. Dumas, Fault attacks on RSA public keys: left-to-right implementations are also vulnerable, in: Proc. The Cryptographers' Track at the Rsa Conference 2009 on Topics in Cryptology, 2009.

[6] J.M. Schmidt, C. A. Herbst, Practical fault attack on square and multiply, in: Proc. The Workshop on Fault Diagnosis and Tolerance in Cryptography, 2008.

[7] A. Barenghi, G. Bertoni, E. Parrinello, Low voltage fault attacks on the RSA cryptosystem, in: Proc. The Workshop on Fault Diagnosis & Tolerance in Cryptography, 2009.

[8] A. Pellegrini, V. Bertacco, T. M. Austin, Fault-based attack of RSA authentication, in: Proc. The Conference on Design, Automation, and Test in Europe, 2010.

[9] M.G. Kuhn, Design principles for tamper-resistant smartcard processors, in: Proc. Usenix Workshop on Smartcard Technology on Usenix Workshop on Smartcard Technology, 1999.

[10] A. Rohani, H.R. Zarandi, An analysis of fault effects and propagations in AVR microcontroller ATmega103 (L), in: Proc. International Conference on Availability, Reliability and Security, 2009.

[11] M. Boreale, Attacking right-to-left modular exponentiation with timely random faults, in: Proc. of the Workshop of Fault

Diagnosis and Tolerance in Cryptography, 2006.

[12] S.M. Yen, M. Joye, Checking before output may not be enough against fault-based cryptanalysis, IEEE Transactions on Computers 49(9)(2000) 967-970.

[13] A. Boscher, R. Naciri, E. Prouff, CRT RSA algorithm protected against fault attacks, in: Proc. International Conference on Information Security Theory and Practices: Smart Cards, Mobile and Ubiquitous Computing Systems, 2007.

[14] P. Bier, W. Fischer, P. Hofreiter, Fault attacks on RSA with CRT: concrete results and practical countermeasures, in: Proc. International Workshop on Cryptographic Hardware and Embedded Systems, 2002.

[15] G. Piret, J. Quisquater, A differential fault attack technique against SPN structures, with application to the AES and Khazad, in: C.D. Walter, Ç. K. Koç, C. Paar, (Eds.), Proceedings of the Cryptographic Hardware and Embedded Systems - CHES 2003, International Workshop, 8-10 September, Cologne, Germany, 2003, pp. 77-88.