Securing RSA Algorithm against Timing Attack

Amuthan Arjunan, Praveena Narayanan, and Kaviarasan Ramu
Department of Computer Science and Engineering, Pondicherry Engineering College, India
Department of Information Technology, Alpha College of Engineering and Technology, India
Department of Computer Science and Engineering, Alpha College of Engineering and Technology, India

Abstract: *Security plays an important role in many embedded systems. All security based algorithms are implemented in hardware or software, and on physical devices which interact with the systems and influenced by their environments. The attacker extracts, investigate and monitor these physical interactions and extracts side channel information which is used in cryptanalysis. This type of cryptanalysis is known as side channel cryptanalysis and attacks performed by using this method is known side channel attacks. There are different types of side channel attacks based on side channel information like time, power, electromagnetic information and faulty output emitted from the cryptographic devices during implementation. The attack that occurs based on the run-time by which the information gained from physical characteristics of cryptosystems to retrieve the secret key is known as the timing attack. The side channel attacks are vulnerable to both symmetric and asymmetric algorithms. RSA is an asymmetric algorithm which plays an important role in most of the applications, but this algorithm is vulnerable to timing attack. So a new technique is proposed called "Randomness Algorithm" and Optical Asymmetric Encryption Padding (OAEP) technique to improve the robustness of RSA algorithm against timing attack, by introducing randomness in computation of decryption process to make the timing information unusable to the attacker.*

Keywords: *Cryptanalysis, side channel attacks, timing attack, RSA, OAEP.*

Received September 13, 2013; accepted March 20, 2014; published online August 22, 2015

1. Introduction

An embedded system is a special-purpose computer system designed to perform a dedicated function. Embedded systems are designed to do some specific task, rather than a general purpose computer which is designed for multiple tasks. Embedded System has software that is embedded into computer hardware, which makes the system dedicated for applications or specific part of an application or part of a larger system. Many modern electronic systems including personal computers, PDAs, cell phones, network routers, smart cards, and network sensors which are used to access, store, manipulate, or communicate sensitive information, making security a serious concern in their design. In embedded system, security is one of the major concerns because the system is vulnerable to both mathematical and implementation attacks. One of the implementation attacks is side channel attacks. Designing a secured cryptographic algorithm in order to overcome the side channel attacks in the embedded system is a challenging task. The weakness of cryptographic algorithm is mainly due to the environmental factors like running, performance optimizations, computations performed during a cryptographic algorithm of embedded systems.

A side channel attack [11] is defined as any observable information which is emitted as a by product of the physical implementation of the cryptosystem. Information leaked via side channels can be any one of the following namely time, power consumption, faulty outputs and electromagnetic radiation of cipher devices. Side channel attacks have been proved to be extremely powerful since they are practical and relatively easier to mount with less constrain due to the complexity of the algorithm [3]. These attacks don't require exhaustive resource and time to launch the attack, which is considered as important threats against modern cryptographic implementations. Side channel attack is more vulnerable to private key as well as public key cryptography which breaks the cryptosystem by inferring the secret key using statistics of phenomena such as timing, power and electromagnetic radiation. Timing attack is a type of side channel attack, in which the attacker attempts to compromise a cryptosystem by analyzing the time taken to execute cryptographic algorithms. The RSA algorithm is used in almost all applications in modern days. RSA is considered as one of the secure public key cryptography which overcomes the mathematical attack and faces security threats in the implementation attack namely timing attack [5]. This attack reveals the secret key of the users and breaks the entire cryptosystem. So, there is need to provide a better solution for mitigating the RSA from timing attack.

The organization of paper is as follows: Section 2 discusses on overview of side channel attack and different types of side channel attack. Section 3 gives a complete description on the techniques used by various researchers to mitigate side channel attack. Section 4 discusses on the attack chosen in RSA algorithm. Section 5 discusses on the proposed technique used to mitigate RSA algorithm. Section 6 discusses on the performance analysis of the proposed

algorithm against timing attack. Conclusion and future work are given in the final section.

The section below discusses on the type of side channel attack.

2. Overview of Side Channel Attack

Side channel attack exploits cryptographic modules information which is extracted from the implementation of the cryptographic primitives and protocols. This characteristic information can be extracted from timing, power consumption or electromagnetic radiation features [3]. These attacks make use of the characteristics of the hardware, software elements as well as the implementation structure of the cryptographic primitive. Therefore, in contrast to analyzing the mathematical structure and properties of the cryptographic primitives, side channel also analyze the implementation features.

Figure 1 represents the hierarchical representation of side channel attack which breaks the implementation of cryptosystem, signature scheme, message authentication and even cryptographic protocols. Some of the known side channel attacks are timing attack, fault attacks, power analysis attack, electromagnetic attack, optical side channel attacks, traffic analysis attacks, acoustic attacks and thermal imaging attacks.

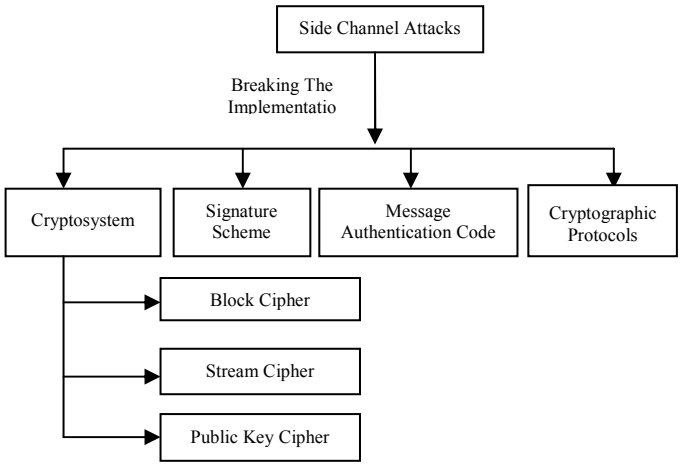

Figure 1. Hierarchical representation of Side channel attack.

2.1. Timing Attack

Timing attack is the attack in which the attacker attempts to compromise a cryptosystem by analyzing the time taken to execute the cryptographic algorithms [6]. Every logical operation in a computer takes time to execute, and that time can differ based on the input.

Implementations of the cryptographic algorithms often perform computations in non-constant time, due to performance optimizations. If such operations involve secret parameters, these timing variations can leak some information and, provided enough knowledge of the implementation which as a form of statistical analysis it could even lead to the total recovery of these secret parameters.

A timing attack is a way of obtaining some user's private information by carefully measuring the time it takes by the user to carry out cryptographic operations. The principle of this attack is very simple and exploits the timing variance in the operation.

2.2. Fault Attack

Fault attack is an active attack which consists of tampering the crypto device in order to create faults and allow them to perform some erroneous operation [2]. This erroneous output will leak information about the secret key used in the crypto device. Two types of faults are used to retrieve the secret key. They are permanent fault and transient fault.

2.3. Power Attack

Power attack is a form of attack in which the attacker studies the power consumption of a cryptographic device. The attack can non-invasively extract cryptographic keys and other secret information from the device [5]. In addition to its running time and its faulty behaviour, the power consumption of a cryptographic device may provide much information about the operations that take place and the involved parameters. Certainly, power analysis attack is applicable only to hardware implementation of the cryptosystems. Power analysis attack is particularly effective and proven successful in attacking smart cards or other dedicated embedded systems storing the secret key.

The below section describes the various techniques used by the researchers to mitigate side channel attack.

3. Survey on Side Channel Attack

Carlos Morino and M. Anwar Hasan [8] proposed the solution for timing attack. In this paper they proposed a counter measure consisting of ideal-weight to make the decryption time in-dependent of the data. The goal of this method is to increase performance penalty when compared to existing blinding method.

Kopf and Durmuth [8] proposed the novel counter measure for timing attack. The amount of information about the key by an unknown message using which the attacker can extract from the deterministic side channel is bounded from above by $|O|log_2(n+1)$ bits where O is set of possible observation, N is the number of side channel measurement. This method leads to implementation with minor performance overhead and formal security guarantees.

Chen *et al*. [4] proposed a technique for mitigating timing attack on RSA, which summarizes several algorithm used to secure RSA implementation like binary algorithm, CRT algorithm, Montogomery reduction Karatsuba multiplication and how timing attack can be used to reconstruct the entire secret RSA exponent. To prevent differences of the running time, the private key operations takes the same amount of time used as a counter measure called as equal timing.

Giraud proposed [5] a technique to counteract with Fault Attack by presenting a new way to implementing exponentiation algorithms. This method can be used to obtain fast FA-resistant RSA signature generations in both the Straightforward Method and Chinese Remainder Theorem modes. The new exponentiation algorithm only adds two modular multiplications and four checksum computations during modular exponentiation of RSA algorithm. This overhead is negligible compared to the cost of the whole exponentiation. Our paper focuses on to protect RSA from timing attack. To overcome the timing attack on RSA proposed two solutions to prevent chosen cipher text attack, time variation and data dependency. First one is padding scheme called Optical Asymmetric Encryption Padding (OAEP) is used before encryption process to prevent chosen cipher text attack and Randomness algorithm is used before decryption process to prevent non fixed time computation in RSA.

From the survey it is quite evident that the techniques discussed is not that much efficient in mitigating side channel attack. So, a new technique is devised in order to make RSA more secure.

4. Problem Undertaken

RSA is asymmetric algorithm most widely used in public key cryptography. The RSA Algorithm was named after Ronald Rivest, Ali and Al-Salami [1], who first published the algorithm in 1977. Since that time, the algorithm has been employed in Internet electronic communications encryption program namely, PGP, Netscape Navigator and Microsoft Explorer web browsers in their implementations of the Secure Sockets Layer (SSL), MasterCard and VISA in the Secure Electronic Transactions (SET) protocol for credit card transactions. The security of the RSA system is based on the intractability of the integer factorization problem. It is very quick to generate large prime numbers using probabilistic algorithms and Rabin-Miller test but very hard to factorise large numbers.

3.1. Modular Exponentiation

Once an RSA cryptosystem is set up, i.e. the modulus n, the private exponent d, public exponent e are determined and the public components have been published, the senders as well as the recipients perform a single operation for encryption and decryption. In RSA encryption or decryption, the core part of the algorithm which takes up much time in the modular exponentiation. Especially in decryption, $M=C^d(mod\ n)$ and since d is generally a big number. For the decryption to run acceptably, speeding up of modular exponentiation is very important.

3.2. Timing Attack Against RSA

In RSA, cryptographic operation in modular exponentiation takes discretely different amount of time to process different inputs. The input (m) and secret key (d) during encryption and decryption process takes different amount of time to execute [5]. Due to this, the attacker observes the operation and notes the time taken during the process and finds the corresponding cipher text or plaintext.

The operation in RSA involves in generating the private key is thus the modular exponentiation $M=C^d\ mod\ N$, where N is the RSA modulus, C is the text to decrypt or sign, and d is the private key. The attacker's goal is to find d. For a timing attack, the attacker needs to compute $C^d\ mod\ N$ for several carefully selected values of C. By precisely measuring the amount of time required and analyzing the timing variations, the attacker can recover the private key d one bit at a time until the entire exponent is known. Add and multiply algorithm is used to perform modular exponentiation of existing RSA.

Timing attack on RSA is mainly due to chosen cipher text attack, non fixed time computation, and data dependency of decryption process. So, analyze the encryption and decryption time for different bits of RSA. From the analysis the encryption time remain constant for different bits but the decryption time may vary depend upon the key size. The variation in key size makes the attacker to guess the value of bits size and break the cryptosystem by retrieve the secret key used in decryption process.

Section 5 depicts the proposed technique to mitigate timing attack in RSA.

5. Proposed Solution

To overcome the timing attack on RSA two techniques are proposed to prevent chosen cipher text attack, time variation and data dependency. First one is padding scheme called OAEP is used before encryption process to prevent chosen cipher text attack and Randomness algorithm is used before decryption process to prevent non fixed time computation in RSA.

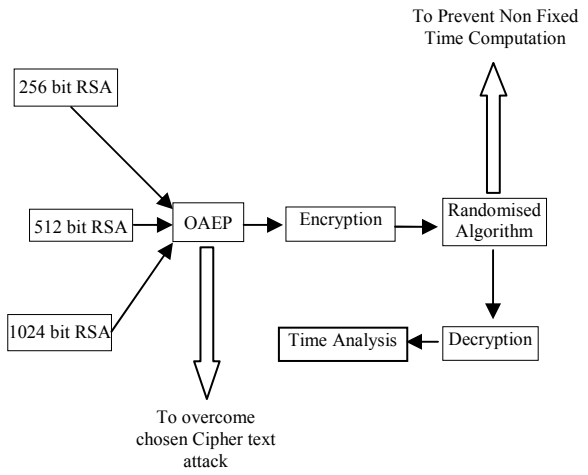

Figure 2. Proposed methods to overcome timing attack.

Figure 2 shows the proposed method consists of two modules OAEP technique and Randomized algorithm used before and after the encryption and the

decryption process to overcome the timing attack for different bits of RSA.

3.3. OAEP Technique

RSA is vulnerable to chosen cipher text attack, to overcome the chosen cipher text attack, padding method called Optimal Asymmetric encryption padding is used. It is a basic Feistel network system and ends up doing a mixture of permuting the plaintext, and adding pseudo random noise to it [11]. It's a reversible transformation and the receiver of the encrypted message knows how to do the reversal of the padding, to decrypt the plaintext. This algorithm uses a pair of random oracles G and H to process the plaintext prior to asymmetric encryption. It increases the size of the messages which guarantees that the encrypted messages are large enough that will not be easy to use for any attack and intersperses pseudo random information that a given Plaintext is encrypted to a wide range of different cipher texts, depending on the choice made during padding.

3.4. Randomized Algorithm

To prevent the non fixed time computation and data dependency randomness is introduced into the computation to make the timing information unusable. This method will work for both encryption and decryption process or even during signature process.

Randomness is introduced into the RSA computations to make timing information unusable. Instead of decrypting the original cipher text $C^d \bmod N$, add new variable A to decryption process and calculate $A^d \bmod N$, before doing the above process calculate the value of A with known values, $A=g^e\ C \bmod N$ where C is cipher text get after the padding method (not original Cipher text) and e is public exponent and g value depends upon the key size of different RSA.

After applying this algorithm in decryption process it produce the decryption time as some blurred time and makes timing information unusable to attacker. Because for any size of RSA like 256, 512 or 1024bit produce the decryption time as blurred time. So from this analysis of timing information the attacker may not even guess the key range of RSA which is basic information to break the cryptosystem.

6. Experimental Results

RSA is implemented using java. Different bits of RSA like 256bit, 512 and 1024bits is implemented, their encryption time and decryption time are analyzed for performing timing attack.

For generating large prime values java.math. BigInteger class is used. The java.math.BigInteger class provides operations analogues to all of Java's primitive integer operators and for all relevant methods from java.lang.Math. It also provides operations for modular arithmetic, GCD calculation, primality testing, prime generation, bit manipulation, and a few other miscellaneous operations. The java.util.SecureRandom class is used to get a cryptographically secure pseudo random number generator used for securing sensitive applications. Depending upon the processor utilized the time for encryption and decryption may vary. But prediction of decryption time is possible due to non fixed computation of the exponential algorithm.

Table 1 represents the existing 256bit implementation of RSA. The prime values of p and q are taken as 128bit size each with key size of maximum of 15 digits and note the encryption as well as decryption time. From the analysis, encryption time remains constant and decryption time vary depends upon the key size.

Table 1. Existing 256bit RSA time analysis.

P (Bit Size)	Q (Bit Size)	Encryption Time (Ms)	Decryption Time(Ms)	Key Size (Digits)
128	128	2	1	5
128	128	2	1	4
128	128	2	2	15
128	128	2	2	8

Table 2 represents the existing 512bit implementation of RSA. The prime values of p and q are taken as 256bit size each with key size of maximum of 15 digits and note the encryption as well as decryption time. From the analysis, encryption time remains constant and decryption time vary depends upon the key size.

Table 2. Existing 512bit RSA time analysis.

P (Bit Size)	Q (Bit Size)	Encryption Time (Ms)	Decryption Time (Ms)	Key Size (Digits)
256	256	5	8	15
256	256	5	7	10
256	256	5	7	8
256	256	5	5	6

Table 3 represents the existing 1024bit implementation of RSA. The prime values of p and q are taken as 512bit size each with key size of maximum of 15 digits and note the encryption as well as decryption time. From the analysis, encryption time remains constant and decryption time vary depends upon the key size

Table 3. Existing 1024bit RSA time analysis.

P (Bit Size)	Q (Bit Size)	Encryption Time (Ms)	Decryption Time (Ms)	Key Size (Digits)
512	512	60	32	15
512	512	60	31	10
512	512	60	30	8
512	512	60	30	6

Table 4 represents the proposed 256bit implementation of RSA. The prime values of p and q are taken as 128bit size with key size of maximum of 15 digits and note the encryption as well as decryption time. From the analysis, encryption time remains constant and produced decryption time is the blurred time to achieve the constant time execution.

Table 4. Proposed 256bit RSA time analysis.

P (Bit Size)	Q (Bit Size)	Enyption Time (Ms)	Blurred Time (Ms)	Key Size (Digits)
128	128	2	1366130282065	5
128	128	2	1366130130299	4
128	128	2	1366130215612	15
128	128	2	1366100789701	8

Table 5 represents the proposed 512bit implementation of RSA. The prime values of *p* and *q* are taken as 256bit size with key size of maximum of 15 digits and note the encryption as well as decryption time. From the analysis, encryption time remains constant and produced decryption time is the blurred time to achieve the constant time execution.

Table 5. Proposed 512bit RSA time analysis.

P (Bit Size)	Q (Bit Size)	Enyption Time (Ms)	Blurred Time (Ms)	Key Size (Digits)
256	256	5	1366131541225	15
256	256	5	1366131711252	10
256	256	5	1366131815939	8
256	256	5	1366131615859	6

Table 6 represents the proposed 1024bit implementation of RSA. The prime values of *p* and *q* are taken as 512bit size with key size of maximum of 15 digits and note the encryption as well as decryption time. From the analysis, encryption time remains constant and produced decryption time is the blurred time to achieve the constant time execution.

Table 6. Proposed 1024bit RSA time analysis.

P (Bit Size)	Q (Bit Size)	Emption Time (Ms)	Blurred Time (Ms)	Key Size (Digits)
512bit	512 bit	60	1366132544562	15
512bit	512 bit	60	1366132156341	10
512bit	512 bit	60	1366132615819	8
512bit	512 bit	60	1366133145623	6

Table 7. Comparing existing time analysis with proposed time analysis of RSA

Bits	Existing System		Proposed System	
	Encryption Time (ms)	Decryption Time (ms)	Encryption Time (ms)	Blurred Time (ms)
256bit RSA	2	1 to 2	2	1366130282065
512bit RSA	5	4 to 6	5	1366131541225
1024bit RSA	60	30 to 32	60	1366132544562

From Table 7 it is noted that in existing system the encryption time remain constant and decryption time vary by increasing the bits of RSA. So there may be chance for attacker to guess the bits of RSA from their time analysis, due to variation in decryption process. But in proposed time it produces the time as blurred time that is fixed time computation for all the bits of RSA, from this timing information it is harder for attacker the to guess the key range. This timing information becomes unusable for attackers as it is not possible to retrieve the secret key *d.*

7. Performance Analysis

The evaluation metrics chosen is time, as timing attack purely depends upon guessing of secret key based on time computed in decryption process using modular exponentiation.

In Figure 3, the decryption time increases if there is increase in key bits size. By using this non fixed time computation as information, attacker have a chance to guess the size of key bits used in decryption process and thereby find the secret exponent *d* by performing the factorization method.

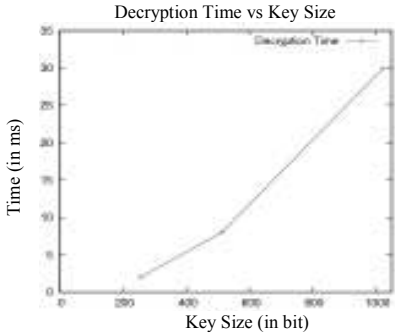

Figure 3. Decryption Time of Existing RSA.

Figure 4 shows the constant time execution of decryption process. The attacker will not guess the key size because of the fixed time computation of all the higher bits of RSA implementation.

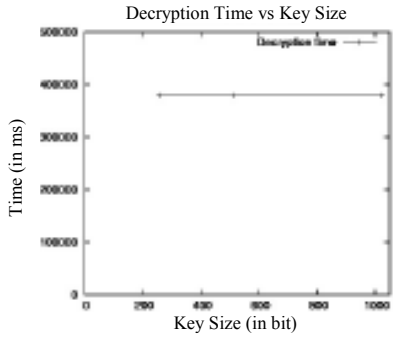

Figure 4. Proposed RSA decryption time.

8. Conclusions

Providing security to the software implementation in embedded systems is one of the challenging techniques. Cryptographic algorithms are always implemented in software or hardware by which interactions is influenced their environments that can be investigated and monitored by attackers and extracts information. Due to which the information leaked due to timing attack makes the secured information revealed. As RSA cryptosystem is currently used in wide variety of applications namely software, hardware (secure telephones, on Ethernet network and smart cards) and many security protocols. The robustness of RSA becomes a question mark among researchers.

So, the technique which is used in this paper will increase the robustness of RSA algorithm against timing attack when compared to existing RSA cryptosystem and possible to make the attacker task harder to find the secret key. As the extension of this work, the same countermeasure can be used for implementing higher bits of RSA and applies the

solution not only for encryption or decryption mode but also for signing and verification mode.

References

[1] Ali H. and Al-Salami M., "Timing Attack Prospect for RSA Cryptanalysts Using Genetic Algorithm Technique," *The International Arab Journal of Information Technology*, vol. 1, no. 1, pp. 80-85, 2004.

[2] Aumuller C., Bier P., Fischer W., Hofreiter P., and Seifert J, "Fault Attacks on RSA with CRT: Concrete Results and Practical Counter-Measures," *in Proceedings of the 4th International Workshop Redwood Shores*, USA, pp. 260-275, 2002.

[3] Borst J, "Block Ciphers: Design, Analysis and Side-Channel Analysis," *PhD Thesis*, K.U.Leuven, 2001.

[4] Chen C., Wang T., and Tian J., "Improving Timing Attack on RSA-CRT via Error Detection and Correction Strategy," *Information Sciences*, vol.232, pp. 464-474, 2013.

[5] Giraud C., "An RSA Implementation Resistant to Fault Attacks and to Simple Power Analysis," *IEEE Transactions on Computers*, vol. 55, no. 9, pp. 1116-1120, 2006.

[6] Kocher P., "Timing Attack on Implementations of Diffie-Hellman, RSA, DSS, and other Systems," available at: http://courses.csail.mit.edu/6.857/2006/handouts/TimingAttacks.pdf, last visited 1996.

[7] Kocher P., Jaffe J., and Jun B., "Differential Power Analysis," available at: https://www.rambus.com/differential-power-analysis/, last visited 1999.

[8] Kopf B. and Durmuth M., "A Provably Secure and Efficient Countermeasure Against Timing Attack," *in Proceeding of the 22nd IEEE Computer Security Foundation Symposium*, Port Jefferson, pp. 324-335, 2009.

[9] Shamir A., "Improved Method and Apparatus for Protecting Public Key Schemes from Timing and Fault Attacks," available at: http://www.google.com/patents/US5991415, last visited 1999.

[10] Srivaths R., Anand R., Kocher P., and Hattangady S., "Security in Embedded Systems: Design Challenges," *ACM Transactions on Embedded Computing Systems*, vol. 3, no. 3, pp. 461-491, 2004.

[11] Zhou Y. and Feng D., "Side-Channel Attacks: Ten Years after its Publication and the Impacts on Cryptographic Module Security Testing," available at: http://eprint.iacr.org/2005/388.pdf, last visited 2005.

Amuthan Arjunan currently, working as Associate Professor in the Department of Computer Science and Engineering, Pondicherry Engineering College, Puducherry. Completed his Under graduate BTech in Computer Science and Engineering from Pondicherry Engineering College, ME from College of Engineering, Anna University, and Chennai. He has obtained his doctorate in the area of Information Security at Pondicherry Engineering College under Pondicherry Universi*ty*.

Praveena Narayanan currently, working as Assistant Professor in Department of Information Technology, Alpha College of Engineering and Technology, Puducherry. She completed BTech in Information Technology from Bharathiyar College of Engineering and Technology and MTech Information Security from Pondicherry Engineering College under Pondicherry University.

Kaviarasan Ramu currently, working as Assistant Professor in Department of Computer Science and Engineering, Alpha College of Engineering and Technology, Puducherry. He completed BTech in Information Technology from Bharathiyar College of Engineering and Technology and MTech Information Security from Pondicherry Engineering College under Pondicherry University.