Crossroads
The ACM Student Magazine

An Introduction to Side Channel Cryptanalysis of RSA

by ***Artemios G. Voyiatzis***

Introduction

Cryptology is the art and science of designing (cryptography) and breaking (cryptanalysis) ciphers based on mathematical tools and researcher creativity. In the middle of the 1990's a new cryptanalysis technique emerged: **implementation cryptanalysis** or **side-channel cryptanalysis**. This technique does not directly attack the strong mathematical constructs of cryptographic algorithms, but rather focuses on implementation details of an algorithm on a physical system. Cryptographic algorithms that withstood years of mathematical cryptanalysis were proven by various researchers to be vulnerable to such attacks, especially in resource-constrained physical systems, such as embedded systems. This article provides an introduction to side-channel cryptanalysis and focuses on side-channel attacks on two implementation variants of the RSA encryption algorithm.

Side Channel Cryptanalysis

Cryptography is a mechanism that provides security in a system or a service, and satisfies data confidentiality and authentication requirements. These two requirements mandate protection of confidential data from outsiders.

On one hand, the implementation of cryptographic algorithms must meet strict requirements because security has been considered an add-on service on an already functioning system. According to this approach, security just consumes valuable

resources. The case for add-on security is even stronger in embedded systems where available resources are limited. Typically, implementation of cryptography focuses on a combination of low power consumption, small memory footprint, high performance, and low latency. In order to meet these requirements, cryptographic algorithm implementations are highly optimized in code size and, in many cases, coded in low level languages like assembly. Various interventions can improve the performance of the algorithm, either with the help of compiler optimizations or even careful inspection and modification of machine code by hand, in order to utilize the available resources more efficiently.

On the other hand, the implementation of an otherwise secure cryptographic algorithm can create side channels. **Side channels** are information channels that carry information about the key or data used. This information leakage is not captured by the mathematical description of the algorithm, but is apparent in a physical implementation of the abstract construct. Examples of such information channels include time variation of the execution of an algorithm, power consumption of the device, or the presence of faulty computations. These channels are more apparent in embedded systems because such systems implement few functions and have small processing and storage capabilities. These system constraints enable an attacker to collect "clearer" information through the side channel in contrast to the "noisy" channel of a general purpose system.

The well-known RSA algorithm [**6**] demonstrates a cryptographic algorithm implementation that does not expose side-channel cryptanalysis threats. The security of the algorithm relies on factoring the product of two large prime numbers (about 512 bits each), $N=p*q$, into its prime factors. Sub-exponential algorithms do exist for factoring such numbers, but their running time complexity is so high that encryption by careful selection of the size of N in bits, denoted as n, is considered quite secure.`

The RSA algorithm's efficiency requires a fast method for performing the **modular exponentiation operation**. A less efficient, conventional method includes raising a number (the input) to a power (the secret or public key of the algorithm, denoted e and d, respectively) and taking the remainder of the division with N. A straight-forward implementation performs these two steps of the operation sequentially: first, raise it to the power and second, apply modulo. This approach is inefficient given the size of the input in bits and conventional processor capabilities. Two more efficient methods are the **Chinese Remainder Theorem** (CRT) and the repeated square and multiply algorithm that utilizes the **Montgomery modular multiplication**. The following demonstrates how these two methods can result in the creation of **side channels**. The information these channels carry can be utilized to derive the secret key of an implementation of the RSA

algorithm.

RSA with Chinese Remainder Theorem

The Chinese Remainder Theorem (CRT) allows for an efficient implementation of the RSA algorithm. The theorem is as follows. Given input, m, raise it to the e-th (or d-th) power modulo p and modulo q. The intermediate results are then combined through multiplication and addition with some predefined constants to compute the final result (the modular exponentiation to N). This approach is often used for implementing RSA in embedded systems. It requires four times less execution time and smaller amount of memory for intermediate results, since modular exponentiation is performed on half the bit size of N.

Boneh, DeMillo, and Lipton [**2**] present an elegant and simple theoretical attack on RSA with CRT implementations. The attack assumes that a computational error occurs during the computation of one of the intermediate results. The faulty result creates an information channel that carries one of the two prime factors of N. For a given message m, it suffices to compute the greatest common divisor of the following two quantities to derive one of the factors: the arithmetic difference between a faulty and a correct signature and the public key N. For example, if the error occurred during the first step of the computation, it holds that $p = GCD(S'-S, N)$

This result comes as a surprise, considering that the RSA algorithm resisted years of pure mathematical cryptanalysis [**1**]. Indeed, the mathematical constructions do not take into account faulty computations, but assume that a computation always results in the correct output. A physical implementation, however, may perform faulty computations. This is a known case, especially for space missions, where the environment can be somewhat hostile. Bit flips can occur for many reasons, such as radiation, power supply or clock manipulation, or even incomplete testing of the hardware during production. While such events might be rare in nature, an attacker can actively force the device to operate in such an environment. Embedded systems can be more vulnerable to such attacks because their resource-constrained environment makes it easy for an attacker to inject a fault in the system. These vulnerabilities are worrisome for financial or commercial applications, such as smart cards for banking (credit cards), cellular phone SIMs, and pay-per-view TV.

Since the introduction of fault-injection attacks, several countermeasures have been proposed. The first proposal performs double computation. In order to detect faulty computations, this proposal computes the result twice before providing an output [**2**]. This is not always efficient because it splits latency and throughput for a given system.

Furthermore, this approach cannot detect permanent faults in which a specific memory area is stuck to a value. The second proposal shows complementary operation of the algorithm. For example, signature verification ensures original input of the algorithm [3]. Signature verification can be rather time-consuming in an embedded system because low value exponents are used for the operation performed by the system and a high value exponent is used by the verifying system.This verification results in higher execution time and complexity.

This field of research has been quite active in recent years. Most, if not all, of the proposed changes in the implementation of the RSA with CRT algorithm, however, were vulnerable to some form of side-channel attack. In many cases, the additional checks and countermeasures inserted in the implementation created additional side channels which can be more easily utilized by attackers instead of protecting the implementation of the algorithm. For a complete treatment of the subject, interested readers should consult [7].

RSA with Repeated Square and Multiply

The original proposal of RSA [6] describes a repeated square and multiply algorithm for fast implementation of the modular exponentiation algorithm. The algorithm segments the expensive modular exponentiation operation in repeated modular squaring and multiplication. The algorithm uses the Montgomery modular multiplication method, which results in even faster implementations [4]. RSA with Montgomery uses the repeated square and multiply algorithm and the Montgomery modular multiplication method for the multiplication step.

The repeated square and multiply algorithm for modular exponentiation works as follows: assume that an RSA encryption must be performed on a message *m* using a key *d* and a public modulus *N* to provide an encrypted message *s*. If d is considered as an array of *k* bits, then the encrypted message can be computed as follows (left-to-right repeated square-and-multiply method):

```
s=1let j = k-1 down to 0s=s*s mod Nif d[j] equals 1     s=s*m mod Nreturn s
```

While RSA with Montgomery can be attacked by fault-injection techniques [2], this paper focuses on passive attack techniques. The implementation, as seen in the code above, contains a conditional execution of a modular multiplication (line 5 of the code), based on the exact value of a key bit (line 4 of the code). This conditional execution creates a side-channel which carries key information that can be utilized by an attacker to derive the secret key. Kocher's timing attack [5] makes three assumptions:

1. A device can be instructed to perform multiple RSA encryptions with the same key,
2. One can measure the execution time of each encryption in the device under attack quite precisely, and
3. The attacker can repeat the measurements on an identical device, where he can set the encryption key at will.

Provided that these assumptions hold, simple statistical analysis of the measurement sets of these two devices can be used to derive the secret key bit-by-bit. The principal idea is to set a key bit on the controlled device and to collect measurements from both devices: the one controlled and the one under attack. Consider the two sets of measurements as random variables. If the key bit on the attacked device matches the one set on the controlled device, then the two random variables are correlated. Otherwise they behave as independent random variables. This differentiation allows the attacker to extract the key bit-by-bit, in linear complexity time.

The same principle is used to derive secret keys, utilizing other passive side-channels such as power consumption or electromagnetic radiation. Such attacks have been also applied to numerous cryptographic algorithms, both symmetric and public-key, which are based on either algebraic structures or elliptical cryptosystems. For a complete treatment on the subject, refer to [7]. In all cases, the differentiation caused by the conditional execution of some instructions based on the key value is exploited in order to derive the secret key.

Passive side-channel attacks and cryptanalysis are harder to detect, because the attacker monitors the behavior of the system and does not actively interact with it. These attacks are also harder to defend because the side channels obey laws of physics and are difficult to harmonize in order to hide the differentiation.

Implementing a Side-channel Attack Resistant Cryptographic Algorithm

The implementation of cryptographic algorithms has not received attention until recently. This was partially caused by the communication gap between system engineers and cryptographers. System engineers usually lack the deep understanding of complexities related to implementing cryptographic algorithms in a secure manner. Instead they focus on meeting vendor requirements where security is typically at the bottom of the list. At the same time, cryptographers tend to focus on the mathematics of cryptography and tend to analyze an algorithm's security in terms of mathematical proofs and algorithmic complexity.

Therefore, side-channel cryptanalysis calls for cooperation and understanding between system engineers and cryptographers. Secure algorithms are vulnerable to simple attacks not described by mathematical models. Yet, cryptographers now understand that information channels can exist in the physical world; such channels are used to apply new or already known cryptanalysis techniques on various algorithms.

Conclusion

The emerging research field of implementation cryptanalysis clearly calls for an increase in education and development of expertise which address both the theoretical and practical issues of implementing cryptographic algorithms in a secure manner.

References

1. Boneh, D. (1999). Twenty years of attacks on the RSA cryptosystem *Notices of the American Mathematical Society (AMS)*, vol. 46, pp. 203-213.
2. Boneh, D., DeMillo, R. A., & Lipton, R. J. (2001). On the Importance of Eliminating Errors in Cryptographic Computations *Journal of Cryptology: The Journal of the International Association for Cryptologic Research*, vol. 14, pp. 101-119.
3. Kaliski, B. & Robshaw, M. J. B. (1997). Comments on Some New Attacks on Cryptographic Devices*. RSA Laboratories Bulletin 5*.
4. Koc, K., Acar, T., & Kaliski Jr., B. S. (1996). Analyzing and Comparing Montgomery Multiplication Algorithms, *IEEE Micro*, vol. 16, pp. 26-33.
5. Kocher, P. C. (1996). Timing Attacks on Implementations of Diffie-Hellman RSA DSS and Other Systems. In *Proceedings of CRYPTO '96*, Lecture Notes in Computer Science, vol. 1109, pp. 104-113. Springer-Verlag.
6. Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystem, *Communications of the ACM*, vol. 21, no. 2, pp. 120-126.
7. Voyiatzis, A. G., Fragopoulos, A. G., & Serpanos, D. N. (2004). Design Issues in Secure Embedded Systems. In *The Handbook of Embedded Systems*, R. Zurawski (ed.). CRC Press.

Biography

Artemios G. Voyiatzis (**bogart@ee.upatras.gr**) is a Ph.D. student in the Department of Electrical and Computer Engineering at the University of Patras in Greece. He is also a student liaison of ACM Crossroads.