MATEC Web of Conferences **76**, 04019 (2016) DOI: 10.1051/matecconf/20167604019
CSCC 2016

Unifier register to protect an efficient modular exponentiation algorithm

David Tinoco Varela[1,a]

[1] *Universidad Nacional Autónoma de México*
Campus Cuautitlán.
Cuautitlán Izcalli, Edo. Mex., México.

Abstract.
Simple power analysis (SPA) attacks are widely used against several cryptosystems, principally against those based on modular exponentiation. Many types of SPA have been reported in the literature in the recent years. There is a real necessity to eliminate the vulnerabilities of cryptosystems, such as CRT-RSA or the Elliptic Curve Cryptosystem, that make them susceptible to these attacks. There are many modular exponentiation algorithms that try to reinforce the security of these systems, of which one was proposed by Da-Zhi et al.

Da-zhi's algorithm was presented as a secure and efficient countermeasure against side channel attacks; however, recently it was shown that its security can be defeated. In this paper, a means of protecting the algorithm is presented. The proposed technique can be applied in any algorithm that computes dummy operations through its execution.

Keywords – Cryptography, modular exponentiation algorithms, side channel attacks, fault attacks, Jacobi symbol.

1 Introduction

A large amount of data are broadcasted daily through electronic communication channels. All of this information must be protected because of the importance and sensitivity of the transmitted data. The best way to protect such information is via high security cryptographic systems.

There are many strong cryptographic systems; two of the most important systems used currently are the RSA cryptosystem, proposed by *Rivest Shamir* and *Adleman* [1]; and the Elliptic Curve Cryptosystem.

A cryptographic system bases its security on many important aspects, such as the mathematical concepts on which it was founded, the computational difficulty needed to obtain the secret keys that ensure data protection, the transmission channel and, in many cases, the use chosen by the final user. Previously, a security system could be considered trustworthy if its mathematical argument was reliable, and the computational difficulty of finding the keys was high; however, this idea was proven false in 1996, when *Kocher* opened the door to a new type of attacks, called the *side channel attacks* (SCA) [2].

SCA are based on a new and different attack methodology that does not depend on formal mathematical concepts or computing power. This new attack scheme is based on observations of the execution time of an electronic device that is running up the modular exponentiation algorithm.

Kocher knew that the execution time of the device could be observed and measured and that from these measurements, it is possible to obtain information as valuable as the secret key of the system. Later, based on this first attack scheme, the extraction of important data from electronic devices by measuring the power consumption of the crypto devices was presented [3].

The measurement of electronic device power consumption at the moment that an algorithm is executed inside of the device created numerous opportunities for physical attacks based on measuring, analyzing, and interpreting any physical signals emanating from the device. Signals such as power consumption, heat emanation, electromagnetic signals, and any other possible signals issued by the device can be measured and used to breach the security of the information systems.

SCA were quickly used to break down the security of cryptosystems based on modular exponentiation (*Add and double* in the elliptic curve cryptosystem), such as RSA, and the method of secure exchange of cryptographic keys *Diffie-Hellman* [4].

[a]Corresponding author: dativa19@hotmail.com

After the SCA, *Bonhe, DeMillo* and *Lipton* presented *Fault attacks* (FA) [5]. FA are more aggressive than SCA because FA physically disturb the execution of the device that is running the cryptographic algorithm.

Many modular exponentiation algorithms have been proposed to avoid these types of physical attacks. The *Square-and-Multiply Always algorithm* developed by *Coron* [6] was the first algorithm specifically designed to avoid such attacks; however, this algorithm was attacked by *Safe error attack* (SEA) [7]. SEA uses dummy operations that the algorithm calculates to break the security of the system. A dummy operation is an operation that is calculated through execution but does not affect the final result of the algorithm's execution.

There are many physical attacks ([8], [9], [10], [11]) trying to break the different modular exponentiation algorithms ([12], [13], [14], [15], [16], [17]) but in 2006 *Boreale* [18] presented a new type of attack, his attack uses a combination between FA and SCA, and according to him, it is possible to get the binary string of the secret key d using the *Jacobi symbol* concept. He attacked the *Square and Multiply Right-to-Left* modular exponentiation and he proved that his attack is effective even in the presence of message blinding. *Schmidt* and *Medwed* [19] used the Jacobi symbol to create an attack which breaks the security of the Montgomery ladder in its blinded form. In the same way *Chong Hee Kim* designed an attack in 2010 [20], based on the Jacobi symbol too, to break the security of the *Add-Only* and *Add-Always* algorithms, both algorithms proposed by Joye in 2007 [21].

Sun Da-Zhi et al. developed an efficient algorithm against *simple power analysis* [22] that uses two binary strings instead of one to calculate modular exponentiation; however, in [23] was presented a way to defeat the security of Da-Zhi's algorithm. The threat was implemented in four steps. It is important to highlight that this assault was designed by observing the behavior of the basic characteristics of Da-zhi's algorithm.

2 Preliminaries

2.1 Quadratic residues and Jacobi symbol

For any prime number p, x is a quadratic residue if $gcd(x,p) = 1$ and $x = y^2 \ mod \ p$ for some y. If $gcd(x,p) = 1$ but x is not a quadratic residue mod p, then x is called a quadratic non-residue modulo p.

In order to illustrate this concept, we will use a modulo $n = 17$. The residues modulo 17 of $1^2, 2^2, ..., 16^2$ are 1, 4, 9, 16, 8, 2, 15, 13, 13, 15, 2, 8, 16, 9, 4, 1; therefore, the quadratic residues modulo 17 are 1, 2, 4, 8, 9, 13, 15, 16.

To determine if a value a is a quadratic residue modulo p, the *Legendre symbol* is used.

$\left(\frac{a}{p}\right)$ is called the *Legendre symbol* of a mod p and has the following properties:

$$\left(\frac{a}{p}\right) = \left\{ \begin{array}{ll} 1 & \text{If } a \text{ is a quadratic residue mod } p \\ -1 & \text{If } a \text{ is a quadratic non-residue mod } p \\ 0 & \text{If there is a common factor} \end{array} \right\}$$

Then, we have that $\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)\cdots\left(\frac{a}{p_k}\right)$ is the Jacobi symbol (JS), where $n = p_1 \cdots p_k$, and p's are prime factors. The Jacobi symbol is a generalization of the Legendre symbol.

2.2 Modular exponentiation

Modular exponentiation is the core of several cryptographic algorithms, and it is also the principal intrusion point of different physical attacks. The classical modular exponentiation algorithm is the *Square and Multiply Right-to-Left*, given as the algorithm 1. This kind of algorithm can be classified into two types: *left-to-right* and *righ-to-left*, depending on the position of the binary string at which the algorithm begins its execution.

Algorithm 1 Square and Multiply Right-to-Left.

```
1: Input $m$, $d = (d_{n-1}, \dots, d_0)_2$
2: Output $s = m^d$
3: $R_0 \leftarrow 1$; $R_1 \leftarrow m$
4: for 0 to $n-1$ do
5:     if $d_i = 1$ then
6:         $R_0 \leftarrow R_0 \cdot R_1 \ mod \ N$
7:     end if
8:     $R_1 \leftarrow R_1^2 \ mod \ N$
9: end for
10: Return $R_0$
```

There are many variations of the modular exponentiation algorithm, all designed to make the security protocol safer.

Chevallier et al. proposed the atomicity technique for modular exponentiation in 2004 [24]. This technique, according to the authors, is a generic scheme and is virtually applicable to any algorithm. The idea says that a process can be seen as a sequence of instructions, and each is equivalent to all other instructions. Hence, they cannot be differentiated through an analysis such as side channel attack; this technique is shown in the algorithm 2.

2.3 Jacobi symbol attacks

As there are many modular exponentiation algorithms, there are also many physical attacks attempting to undermine the security of the exponentiation algorithms.

Algorithm 2 SaM protected by the atomicity technique.

```
 1: Input $m, d, N$, with $d = (d_{n-1} \cdots d_0)_2$
 2: Output $S = m^d \ mod \ N$
 3: $R[0] \leftarrow 1$
 4: $R[1] \leftarrow m$
 5: $k = 0$
 6: $i = n - 1$
 7: while $i \geq 0$ do
 8:     $R[0] \leftarrow R[0] \cdot R[k] \ mod \ N$
 9:     $k \leftarrow k \oplus d_i$
10:     $i \leftarrow i - \bar{k}$
11: end for
12: Return $R[0]$
```

After the apparition of SCA and FA, attacks that used one of these two techniques but were also based on a combination of mathematical and numerical concepts were developed, such as the attack proposed in [8].

Other types of attacks that combine physical attacks with numerical concepts use the Jacobi symbol to defeat cryptographic security.

Attacks using the Jacobi symbol began with *Boreale*, when he attacked the binary Square and Multiply Right-to-Left modular exponentiation (Algorithm 1) in 2006 [18]. He puts a fault z in R_1 when a squaring is executed in iteration $i - 1$ of the Square-and-Multiply algorithm, and then, depending on the calculation of (S/N), where S is the output value of the algorithm and N is the modulus, it is possible to know what value of the bit d_i was attacked. This scheme works by assuming that $\left(\frac{m}{N}\right) = 1$, and its behavior is similar to the *Safe error*: if the bit in iteration i is equal to 0, the fault does not affect the calculation of the JS of (R_{0_i}/N), and z is squaring, so that $(z^2/N) = 1$, but if $d_i = 1$, z affects the register R_{0_i} and can have the JS value $(R_{0_i}/N) = -1$. Then, z can be or not be a quadratic residue. If z is a quadratic residue, then the final result will be $(S/N) = 1$, but if it is a quadratic non-residue, the final result will be $(S/N) = -1$. For this reason, his attack is a probabilistic model.

After Boreale, *Schmidt* [19] proposed an attack that entailed sending a message m with $(m/N) = -1$ to the *Fumaroli-Vigilant algorithm* [25] and skipping the operation $R_{d_i} = {R_{d_i}}^2$ in the algorithm. By observing the JS of the resulting value, it is possible to learn about the values of d_i and d_{i-1}. If (S/N) is equal to -1, then $d_i = d_{i-1}$.

The attacks mentioned above are easy to implement, and they are powerful because they require only the JS in the returned value by the attacked algorithm to breach the security of a cryptosystem. However, a way to avoid this kind of attack was shown in [26].

2.4 Attack on Da-zhi's algorithm

Sun Da-Zhi et al. published a modular exponentiation algorithm that separates the original binary string of the exponent d into two binary strings d_1 and d_2. The algorithm is given as algorithm 3, where $sq^{(y)}(A)$ means performing y modular squares on the integer A [22].

Algorithm 3 Sun Da-zhi's algorithm [22].

```
 1: Input $m, d, N$ with $d_1 = d_{\lceil k/2 \rceil,1} \cdots d_{1,1}$ and $d_2 =$
    $d_{\lceil k/2 \rceil,2} \cdots d_{1,2}$
 2: Output $C = m^d \ mod \ N$
 3: $s = m$
 4: $C_0 = C_1 = C_2 = C_3 = 1;$
 5: $C_{2 \cdot d_{1,2} + d_{1,1}} = s \cdot C_{2 \cdot d_{1,2} + d_{1,1}}$
 6: for 2 to $\lceil k/2 \rceil$ do
 7:     $s = s \cdot s$
 8:     $C_{2 \cdot d_{i,2} + d_{i,1}} = s \cdot C_{2 \cdot d_{i,2} + d_{i,1}}$
 9: end for
10: $C_1 = C_1 \cdot C_3$
11: $C_2 = C_2 \cdot C_3$
12: $C = sq^{(\lceil k/2 \rceil)}(m^{C_1}) \cdot m^{C_2}$
```

The key idea in algorithm 3 is to separate the k-bit binary string d into two ($\lceil k/2 \rceil$)-bit binary strings d_1 and d_2, and depending on the values of d_1 and d_2 in each iteration, four registers $C_0 \cdots C_3$ can be utilized to calculate the exponentiation. If $d_{i,1}$ and $d_{i,2}$ (Where $i, 1$ is the i-bit of d_1 and $i, 2$ is the i-bit of d_2) are equal to 0, the chosen register is C_0; if $d_{i,1} = 1$ and $d_{i,2} = 0$, the chosen register is C_1, and so on, for each register.

Algorithm 3 was presented as a secure algorithm against SCA. However in [23] was demonstrated that the security of the algorithm can be defeated using an attack consisting of four steps. The attack is described below:

1. When $d_{i,1} = d_{i,2} = 0$, register C_0 is used. Every time this register is picked, Da-zhi's algorithm executes a dummy operation. If an attacker performs a FA when C_0 is selected, then the final result will not be affected. Thus, an attacker can determine the bits combination relative to the point attacked; thus, he can find a quarter of the total binary string.

2. We can see at line 12 from algorithm 3, that the register C_1 is always squared. Therefore, if any FA is put over that register and alters its JS, Jacobi symbol value in the final calculation will always become 1 because all negative JS change to a positive JS due to the operation $C = sq^{(\lceil k/2 \rceil)}(m^{C_1})$. Thus, by performing several FA and studying the behavior of the returned value, an attacker can easily identify where the Register C_1 was used; therefore, he can know one of four bits combinations that the algorithm uses.

3. To use the register C_2, it is necessary that $d_{i,1} = 0$ and $d_{i,2} = 1$. To use the register C_3, it is necessary that $d_{i,1} = 1$ and $d_{i,2} = 1$. It can be noted that for both registers, the value of the bit $d_{i,2}$ in i will always be equal to 1; therefore, it is possible to determine that all of the unknown values of the bits in d_2 are equal to 1.

4. One quarter, on average, of the values of the binary string d are unknown, but these can be found via different algorithms in the literature[27].

3 Protecting Da-zhi's algorithm using a unifier register

As described previously, the first step in attacking the Da-zhi's algorithm is to determine the position of the exponent's binary string in which a dummy operation is executed. To avoid this situation, it is necessary to unify all of the algorithm's registers $C_0...C_3$ to prevent a dummy operation from being used.

We realized that using the register C_0 in an independent manner can permit the unification of the registers; this value can later be allocated in any other register used by algorithm 3. Due to the necessity of making the C_0 register independent, a first approach to solving this problem involved the use of the technique called *atomicity*, proposed in [24]. As mentioned previously, this technique requires that each iteration of an algorithm is equal in time and power consumption to the others. The implementation of this concept alone does not solve the problem, but it can allow us to work with the C_0 register separately.

In algorithm 4, we can see the implementation of the atomicity technique over the Da-zhi's algorithm.

Algorithm 4 Da-zhi's algorithm with atomicity.

```
1: Input $m, d, N$ with $d_1 = d_{\lceil n/2\rceil,1} \cdots d_{1,1}$ and $d_2 = d_{\lceil n/2\rceil,2} \cdots d_{1,2}$
2: Output $C = m^d \ mod\ N$
3: $C_0 = m$
4: $C_1 = C_2 = C_3 = 1$;
5: $i = k = 1$;
6: while $i \leq \lceil i/2\rceil$ do
7:     $k = k \oplus [d_{i,1}\ OR\ d_{i,2}]$;
8:     $C_{2kd_{i,2}+kd_{i,1}} = C_0 \cdot C_{2kd_{i,2}+kd_{i,1}}$;
9:     $i = i + k$;
10: end while
11: $C_1 = C_1 \cdot C_3$
12: $C_2 = C_2 \cdot C_3$
13: $C = sq^{(\lceil n/2\rceil)}(C_1) \cdot C_2$
```

Algorithm 4 uses all of the registers to correctly calculate of the exponentiation result. Thus, it does not matter which is the altered register as the final result will always be inaccurate. This is an advantage over the original algorithm because in the original algorithm, if one fault was placed at C_0, the final result was unaltered. It should also be noted that the C_0 register is executed in each iteration of algorithm 4, which is an important feature of this algorithm. Despite its characteristics, this algorithm is insecure against Jacobi symbol attacks. This vulnerability will be explained in the next section.

After a way was found to manipulate C_0 in an independent form, algorithm 4 was modified to make it safe against FA and JS attack, thus obtaining the algorithm 5, where $\oplus$ represents the binary operation OR exclusive. This algorithm avoids the use of dummy operations through an interconnection of all the registers used in its execution, meaning that a dummy operation is converted into an operation that alters the final result of the calculation.

Algorithm 5 Da-zhi's algorithm with atomicity and the unifier register.

```
1: Input $m, d, N$ with $d_1 = d_{\lceil n/2\rceil,1} \cdots d_{1,1}$ and $d_2 = d_{\lceil n/2\rceil,2} \cdots d_{1,2}$
2: Output $C = m^d \ mod\ N$
3: $C_0 = m$
4: $C_1 = C_2 = C_3 = C_4 = 1$;
5: $i = k = b = 0$;
6: while $i \leq \lceil i/2\rceil$ do
7:     $k = \bar{k}\ AND\ (b \oplus [d_{i,1}\ OR\ d_{i,2}])$;
8:     $b = k\ OR\ \bar{b}$;
9:     $C_{3k+b-2kd_{i,2}-kd_{i,1}} = C_0 \cdot C_{3k+b-2kd_{i,2}-kd_{i,1}}$;
10:    $i = i + k$;
11: end while
12: $C_1 = C_1 \cdot C_4^{-1}$;
13: $C_3 = C_3 \cdot C_1$
14: $C_2 = C_2 \cdot C_1$
15: $C = sq^{(\lceil n/2\rceil)}(C_3) \cdot C_2$
```

Algorithm 5 has a higher number of registers than algorithm 3, including the C_4 register. However, the increase in registers is minimal because this modification will provide security for any information system in which this idea can be implemented. In other words, a minimum amount of memory is sacrificed, but a guarantee of security will be obtained.

Now, we will see how the algorithm 5 avoids the dummy operation. When a fault is allocated into C_0, the fault will modify the value of this register. Such erroneous values always disturb the final result. To obtain the general disturbance, C_0 will multiply to C_1 and C_4 when $d_{i,1} = d_{i,2} = 0$. Therefore, C_0 will not be multiplied by itself anymore (the importance of this fact will be described in the next section). This means that when an FA is allocated into C_0, this erroneous value will join another register through the algorithm's execution. As a result, it will be altering all of the necessary values involved in the calculation of the exponentiation. Thus, C_0 becomes a *unifier register*. Registers can be unified, and the erroneous value in C_0 will disturb the final result of the operation, thereby eliminating the dummy operations of the algorithm's execution.

C_4 register is used to contain a multiplicative inverse of C_0, which will eliminate the C_0 value from C_1 at line 12 of algorithm 5. We can also see that only one register is protected with the value of C_0 and so, it was not necessary to protect all of the remaining registers.

It is easy to see that the register selection is different than that of the algorithm 3. In this algorithm, when $d_{i,1} = d_{i,2} = 0$, the registers C_1 and C_4 are used; when $d_{i,1} = 1$ and $d_{i,2} = 0$, C_3 is used; when $d_{i,1} = 0$ and $d_{i,2} = 1$, C_2 is used; and finally, when $d_{i,1} = 1$ and $d_{i,2} = 1$, C_1 is used. C_0 is executed in all the bits combinations. First, C_0 is calculated. Then, the register corresponding to the combination is used.

This algorithm is only slightly more complex than the original one. The first step in its design was to determine the expressions to calculate b and k to make the algorithm performs three multiplications when $d_{i,1} = d_{i,2} = 0$ and two multiplications in any other case. Both formulas were obtained with the help of *Karnaugh maps*.

The unifier register is not limited to the Da-zhi's algorithm. It can be used in any algorithm that makes dummy operations through its execution to avoid FA.

4 Analysis of the displayed algorithm

The Jacobi symbol has been used to attack many exponentiation algorithms. The general form of the attack consists of putting an FA through the algorithm's execution, calculating the JS of the obtained result, verifying the JS values 1 and -1, repeating the same attack many times, and comparing the obtained JS from those attacks. Thus, an attacker can obtain the secret key's binary string.

To verify the security of Da-zhi's algorithm modified with the atomicity technique (algorithm 4) and to compare it with our algorithm implemented with the unifier register (algorithm 5), Jacobi symbol attacks were made over them. The attack is described below.

When we performed safety tests of the algorithm 4 with respect to the JS, we realized that it was effective against common FA; however, it allows attacks by the JS concept. If a fault is placed in C_0 when $d_{i,1} = d_{i,2} = 0$, the next operation in the iteration $i + 1$ of the algorithm is $C_0 = C_0 \cdot C_0$. Hence, if the erroneous value had a JS equal to -1 in i, in the iteration $i + 1$ the JS will be changed to 1. The other three combinations of $d_{i,1}$ and $d_{i,2}$ do not exhibit this behavior because the $JS = -1$ of C_0 always affects one of three registers in the next step such that $C_{2kd_{i,2}+kd_{i,1}} = C_0 \cdot C_{2kd_{i,2}+kd_{i,1}}$ (where $2kd_{i,2} + kd_{i,1} \neq 0$, for this case). Based on these data, we can determine that if a fault was placed when $d_{i,1} = d_{i,2} = 0$, the algorithm will always have a $JS = 1$ in the output value.

Although the implementation of the atomicity technique over Da-zhi's algorithm is very useful and witty, it has vulnerabilities against attacks that use the JS. The algorithm implemented with the unifier register (algorithm 5) does not present vulnerabilities against these types of attacks because it does not perform the operation $C_0 = C_0 \cdot C_0$. Thus, a negative JS in C_0 is unified with others registers. In the same manner that an FA can be propagated through the algorithm's execution, a JS value can be propagated through the calculations, and can disturb any final result (as explained previously). This avoids the identification of any specific bits combinations as all of them alter the JS of the final result. For this reason, an algorithm with unifier register has a better security level.

Because the algorithm has an atomized behavior, it is a regular algorithm and is secure against simple side channel attacks.

The version of Da-zhi's algorithm presented here has different characteristics than the original. Table 1 shows a summary of the characteristics of such algorithms.

In table 1 the multiplications and the squaring are considered a multiplication. In the column "number of variables" d_1, d_2, k, i, m, d and N are not counted because they are variables inherent in the modular exponentiation and are not related to their implementation.

Many modular exponentiation algorithms execute more calculations, whereas the binary form of the exponent has more 1's. For this reason, binary representations with less 1's are sought. *Non adjacent form* (NAF) [28] is a signed binary representation that can be used to reduce the quantity of 1's inside of a binary string. NAF has two main characteristics: First, the quantity of 1's in any binary string is $n/3$ on average, where n is the bit length. Second, this form has a bit length of $1 + n$ bits. We can see this behavior in the next example:

Decimal	**Binary**	**NAF**
98274	10111111111100010	10-1000000000-100010

The last characteristic can be a disadvantage when considering that many crypto devices that execute modular exponentiation algorithms are designed to work with a specific exponent's bit length (secret key). This means that they are designed to operate with 1024 bits, not 1025. In other words, an electronic device that executes a cryptographic algorithm would have to be redesigned to work properly with signed binary representations, a very expensive situation.

Table 1. Caracteristichs of the modified algorithms.

Algorithm	**Number of multiplications**	**Inversions**	**Number of Variables**
Algoritmo 3	$1.5n + 2$	0	5
Algoritmo 4	$1n + (3/8)n + 3 = 1.375n + 3$	0	4
Algoritmo 5	$1.5n + (1/8)n + 4 = 1.625n + 4$	1	6

Thus, it is important to note that the modified algorithm by the unifier register is not compatible with signed binary representations, such as NAF. As stated previously, the algorithm 5 makes three operations when the bit combination is $d_{i,1} = d_{i,2} = 0$. Therefore, while more bits equal to 0 exist, more operations are performed, and their execution times are longer. In this case, it is not necessary to search alternate binary representations that minimize the quantity of 1's. This can be considered an advantage over other algorithms because it can work more efficiently with unsigned binary representations. Hence, it can be in concordance with crypto devices that use standard bit lengths to function.

5 Conclusions

In this work we have presented a way to protect algorithms in right-to-left form that use dummy operations in their execution. When the unifier register is used, the algorithm's security is greater than that in the algorithms that are not protected. The protected algorithms can be incompatible with signed binary representations, such as NAF. This means that they are more efficient when the binary string has few bits equal to 0. However, this characteristic can be seen as an advantage if the implementation over crypto devices that use standard key bit lengths to execute an encryption algorithm is considered. Therefore, it is not necessary to change the hardware to use the unifier register, which may be a very important property.

In the specific case of the Da-zhi's algorithm, this simple modification avoids that the bits of the binary string which represents the secret key can be obtained as described by [23].

Additionally, this algorithm is secure against simple side channel attacks due to atomicity.

Acknowledgments

The authors thank PAPIIT IN 113316, PIAPI1634, and Program PCIC of the UNAM.

References

[1] RL Rivest, A. Shamir and L. Adleman. "A method for obtaining digital signatures and public-key cryptosystems". *Communications of the ACM*, **21**(2):120-126 (1978).

[2] P. Kocher. "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems". *In Koblitz, N., ed.: Advances in Cryptology-CRYPTO 96*, **1109** of Lecture Notes in Computer Science:104-113, Springer (1996).

[3] P.C. Kocher, J. Jaffe and B. Jun. "Differential Power Analysis". *In Wiener, M., Ed.: Advances in Cryptology-CRYPTO '99.* **1666** *of Lecture Notes in Computer Science. Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*: 388-397, Springer (1999).

[4] Whitfield Diffie and Martin E. Hellman. "New directions in cryptography". *IEEE Transactions on Information Theory*, **22**(6):644-654, IEEE (1976).

[5] D. Boneh, R. DeMillo and R. Lipton. "On the importance of checking cryptographic protocols for faults". *In Fumy, W., Ed.: Advances in Cryptology-EUROCRYPT'97.* **1233** *of Lecture Notes in Computer Science*, pages 37-51, Springer (1997).

[6] J.S. Coron. "Resistance against differential power analysis for elliptic curve cryptosystems". *In Koç, Ç., Paar, C., Eds.: Cryptographic Hardware and Embedded Systems-CHES 2002.* **1717** *of Lecture Notes in Computer Science*, pages 292-302, Springer (1999).

[7] S.M. Yen, S. Kim, S. Lim and S. Moon. "A countermeasure against one physical cryptanalysis may benefit another attack". *Information Security and Cryptology-ICISC 2001*, **2288** of Lecture Notes in Computer Science:414-427, Springer (2001).

[8] S.M. Yen, W.C. Lien, S.J. Moon and J.C. Ha. "Power analysis by exploiting chosen message and internal collisions–vulnerability of checking mechanism for RSA-decryption". *Progress in Cryptology–Mycrypt 2005*, **3715** of Lecture Notes in Computer Science:183-195, Springer (2005).

[9] C.H. Kim and J.J. Quisquater. Fault attacks for CRT based RSA: New attacks, new results, and new countermeasures. *Information Security Theory and Practices. Smart Cards, Mobile and Ubiquitous Computing Systems*, 4462:215-228, Springer (2007).

[10] S. Chari, J. Rao and P. Rohatgi. "Template attacks". *Cryptographic Hardware and Embedded Systems-CHES 2002*, **2523** of Lecture Notes in Computer Science:12-28, Springer (2002).

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

[12] H. Mamiya, A. Miyaji and H. Morimoto. "Efficient countermeasures against RPA, DPA, and SPA".

Cryptographic Hardware and Embedded Systems-CHES 2004, **3156** of Lecture Notes in Computer Science:343-356, Springer (2004).

[13] C.C. Lu, S.Y. Tseng and S.K. Huang. "A secure modular exponential algorithm resists to power, timing, C safe error and M safe error attacks". *In 19th International Conference on Advanced Information Networking and Applications, 2005. AINA 2005*, **2**, pages 151-154, IEEE (2005).

[14] C.H. Kim and J.J. "Quisquater. How can we overcome both side channel analysis and fault attacks on RSA-CRT?". *Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 21-29, IEEE (2007).

[15] A. Boscher, R. Naciri and E. Prouff."CRT RSA algorithm protected against fault attacks". *Information Security Theory and Practices. Smart Cards, Mobile and Ubiquitous Computing Systems*, **4462** of LNCS: 229-243, Springer (2007).

[16] J.C. Ha, C.H. Jun, J.H. Park, S.J. Moon and C.K. Kim. "A New CRT-RSA Scheme Resistant to Power Analysis and Fault Attacks". *Third 2008 International Conference on Convergence and Hybrid Information Technology*:351-356, IEEE (2008).

[17] A. Boscher, H. Handschuh and E. Trichina. "Blinded fault resistant exponentiation revisited". *In L. Breveglieri, S. Gueron, I. Koren, D. Naccache, and J.-P. Seifert, editors, Workshop on Fault Diagnosis and Tolerance in Crptography - FDTC'09*, pages 3-9, IEEE (2009).

[18] M. Boreale. "Attacking right-to-left modular exponentiation with timely random faults". *Fault Diagnosis and Tolerance in Cryptography*, **4236** of LNCS: 24-35, Springer (2006).

[19] Jörn-Marc Schmidt and Marcel Medwed. "Fault Attacks on the Montgomery Powering Ladder". *13th Annual International Conference on Information Security and Cryptology, Proceedings*, LNCS, Springer (2010).

[20] C.H. Kim. "New fault attacks using Jacobi symbol and application to regular right-to-left algorithms". *Information Processing Letters*, **110**(20):882-886, Elsevier (2010).

[21] M. Joye."Highly regular right-to-left algorithms for scalar multiplication". *Cryptographic Hardware and Embedded Systems-CHES 2007*, **4727** of Lecture in Notes in Computer Science:135-147, Springer (2007).

[22] D.Z. Sun, J.P. Huai, J.Z. Sun and Z.F. Cao. "An efficient modular exponentiation algorithm against simple power analysis attacks". *Consumer Electronics, IEEE Transactions on*, **53**(4):1718-1723 (2007).

[23] D. Tinoco Varela. "How to break down the security of an efficient modular exponentiation algorithm". *Recent advances in computer science*, Proceedings of the 19th International Conference on Computers: 81-85, INASE (2015).

[24] B. Chevallier-Mames, M. Ciet and M. Joye. "Low-cost solutions for preventing simple side-channel analysis: Side-channel atomicity. *IEEE Transactions on Computers*, **53**(6): 760-768, IEEE (2004).

[25] G. Fumaroli and D. Vigilant. "Blinded fault resistant exponentiation". *Fault Diagnosis and Tolerance in Cryptography*, **4236** of Lecture Notes in Computer Science:62-70, Springer-Verlag (2006).

[26] D. Tinoco Varela. "Blinded Montgomery Powering Ladder Protected Against the Jacobi Symbol Attack". *International Journal of Security (IJS)*, **6**(3), 15-27 (2012).

[27] Johannes Blömer and Alexander May. "New partial key exposure attacks on RSA". *Advances in Cryptology-CRYPTO 200*, **3**:27–43, Springer (2003).

[28] G.W. . Reitwiesner. "Binary arithmetic" . *Advances in computers*, **1**: 231-308, Elsevier (1960).