Countermeasures for Hardware Fault Attack in Multi-Prime RSA Cryptosystems

Zine-Eddine Abid[1] and Wei Wang[2]
(Corresponding author: Z. Abid)

Department of Electrical and Computer Engineering, The University of Western Ontario[1]
1151 Richmond St, London, Ontario, Canada, N6A 5B9 (Email: zeabid@eng.uwo.ca)
Indiana University-Purdue University at Indianapolis, Indianapolis, Indiana, USA[2]

(Received Apr. 4, 2006; revised and accepted July 31, 2006)

Abstract

The study of countermeasures for hardware fault attack in multi-prime RSA cryptosystems is very important for applications such as computer network and smart cards. In this paper, an efficient countermeasure method is proposed for the FPGA-based multi-prime RSA systems. The proposed method can survive the attacks [27, 30] that broke the previous methods [5, 33]. Furthermore, by using a simple operation and small wordlength parameters, the proposed method is very efficient in terms of hardware resources and speed. In order to verify the effectiveness of the proposed method, the FPGA implementation and testing in attacking environment are carried out for several two-prime and three-prime design examples.

Keywords: Chinese remainder theorem, countermeasure, FPGA, hardware fault attack, immunity, RSA cryptosystem

1 Introduction

With the rapid growth of the Internet and telecommunication systems, the security of cryptosystems is becoming more and more important [9, 22]. The study of various attacks and their corresponding countermeasures for the cryptosystems has drawn a lot of attentions. The most popular ones are power, timing and fault attacks, which can be classified into side channel attacks [1, 2, 3, 5, 6, 8, 9, 12, 13, 14, 19, 20, 22, 24, 25, 27, 29, 30, 31, 32, 33, 34, 35]. Amongst different side channel attacks, hardware fault attack [2, 6, 8, 19] seems to be the easiest to realize and is proved to be effective in breaking the implementation of Chinese Remainder Theorem (CRT)-based RSA systems, namely, multi-prime RSA or CRT-RSA. Since a multi-prime RSA can speed up large modular operations of RSA and is very useful for applications such as smart cards, the cryptanalysis and countermeasure study on hardware fault attack for a multi-prime RSA is of high importance. When a fault attack occurs, the public modulus of the cryptosystem may be factored due to some faulty computation values purposely induced by the attacker. There has been extensive study in the literature [2, 3, 5, 6, 8, 12, 19, 25, 27, 29, 30, 31, 32, 33, 34, 35] to explore securing multi-prime RSA against hardware fault attacks, so that the CRT's speed advantage is retained without its susceptibility to fault attacks. One countermeasure prevents the attacker from accessing the message by using hash function [12]. Another one is Shamir's checking method, which contains checking the interim calculation results to guarantee the correctness of the signature [25]. To counteract hardware fault attack, Shamir's checking method is sometimes not very effective, since countermeasures without checking procedure are desirable for RSA cryptosystems [32, 34, 35]. Yen et. al. [33] proposed non-checking protocols (CRT-1 and CRT-2) based on an appealing notion of infective computation. This is a method where any error introduced by a fault propagates throughout the computation, ultimately ensuring the final result to be randomized and not to be used to break the system. Consequently, with an infective algorithm, there is no need to check the output for errors. However, these protocols proposed by Yen, et. al. were later shown to be insecure in a new fault attack [30]. At ACM CCS 2003, Blomer et. al, proposed another scheme, namely, BOS algorithm, also based on the ideas of infective computation [5]. However, it was demonstrated in [27] that the BOS algorithm can also be broken. Therefore, it remains a challenging open problem to find an infective implementation of multi-prime RSA. To ensure security in practice, one must consider the properties of different implementations of cryptosystems. In this work, we focus the countermeasure study against hardware fault attack for the FPGA implementation of the multi-prime RSA. One advantage of the FPGA implementation of such a system is that logic cells and interconnects can be easily scattered, which limits the ability of the attacker to control the timing and location of the computation [8]. Therefore, an infective multi-prime RSA can be achieved

for FPGA implementation. In this paper, a new countermeasure method is proposed for FPGA-based multi-prime RSA cryptosystems, which can resist hardware fault attack. The calculation expression of the standard RSA signature is revised by including an additive term to ensure that no useful information can be obtained from the faulty output signature. The proposed method can survive the attacks [27, 30] that broke the previous methods [5, 33]. By using a simpler operation and smaller size parameters, the proposed method is also very efficient in terms of hardware resources and speed. In order to prove the functions of the proposed method, the FPGA implementation and testing in attacking environment are carried out for several two-prime and three-prime design examples.

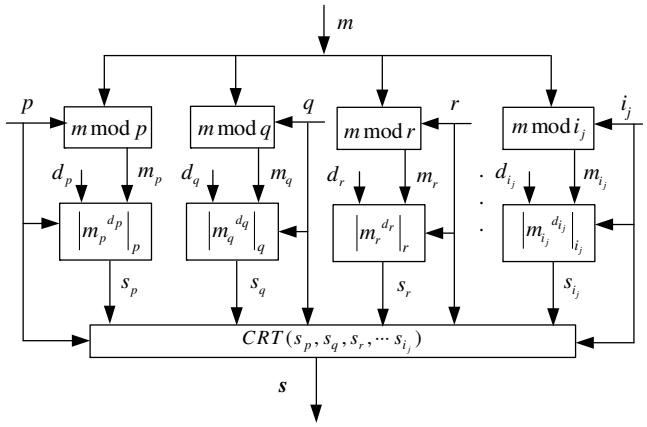

Figure 1: The block diagram of the signing process for a multi-prime RSA

2 Multi-prime RSA and Hardware Fault Attack

RSA [22], as a popular public key cryptography, is widely used to provide the essential security over the network [9]. RSA can be used for both encryption/decryption and signing process. In this paper, our study concentrates on the signing process of RSA systems, which computes digital signatures.

2.1 Basics of Multi-prime RSA

The basic RSA cryptosystem has two public quantities referred to as n (modulus) and e, and two private quantities d and $\lambda(n)$. $\lambda(n)$ is defined as the Least Common Multiple (LCM) of the factors of n. The public exponent e is an integer smaller than $\lambda(n)$, which satisfies gcd $(e, \lambda(n)) = 1$. The private key d is calculated as $d = e^{-1} \bmod \lambda(n)$. During the signing and signature-verification process, the secret key d is used to obtain the signature of message m as $s = m^d \bmod n$ and the public key is to verify the signature by checking whether $s^e \bmod n$ equals to the message m.

The main computation of RSA is based on a modular exponentiation of large integers, which involves long delay and many hardware resources. In order to speed up RSA cryptosystem, a multi-prime RSA is generally used based on Chinese Remainder Theorem, in which an RSA operation is replaced by several parallel operations with smaller bases and exponents [7, 4, 10, 23, 28]. This speed gain is very useful for applications such as smart cards.

A multi-prime RSA cryptosystem is shown in Figure 1. The modulus n has multiple prime factors $i_1, \cdots, i_j$, such that $n = \prod_{k-1}^{j} i_k$. The quantities $\{n, e\}$ are made public while $\{i_1, \cdots, i_j, \lambda(n), d\}$ are kept private. Then, the signing process can be accelerated by using a general j-prime RSA as follows. For $k = 1, \cdots, j$, we have

$$\begin{aligned} m_i &= m \bmod i_k \\ s_{i_k} &= m_{i_k}^{di_k} \bmod i_k \quad \text{where} \quad d_{i_k} = d \bmod (i_k - 1) \\ s &= CRT(s_{i_1}, \cdots, s_{i_j}), \end{aligned} \tag{1}$$

where j is the number of prime factors in the RSA system.

The CRT equation is denoted as follows [6]. An integer p is defined as $p = \prod_{k=1}^{j} p_k$, where p_k's are positive integers and relatively prime in pairs (here, $p_i \neq p_j$, and if $i \neq j$, all $p_1, p_2, \cdots, p_j$ are primes). Also, we have $x_k = x \bmod p_k$. Then, x can be calculated by $x = (\sum_{k=1}^{j} x_k r_k s_k) \bmod p = CRT(s_1, \cdots, s_j)$ where $k = 1, \cdots, j, r_k = \frac{p}{p_k}$ and $s_k = r_k^{-1} \bmod p_k$.

It is clear that a large public modulus n is split into j small wordlength prime numbers in the CRT algorithm. Thus, the modular exponentiation operation of the basic RSA is calculated in parallel using smaller numbers, thus achieving high speed. If the modulus is a product of j primes of equal length, the multi-prime RSA has a speed up of j^2.

It is also noted that the multi-prime RSA with more channels or a large value of j provides higher operating speed than the one with less channels or a small j. For example, a three-prime RSA gives a theoretical speedup by a factor of 9/4 compared to the corresponding two-prime case [6]. However, due to the implementation efficiency, the most popular used ones are generally two-prime or three-prime RSAs.

2.2 Hardware Fault Attack

For a multi-prime RSA cryptosystem, the attacker is able to purposely introduce some types of hardware fault into the system. Then, the affected computation values may be used to factor the public modulus of the system [1, 20]. This is called hardware fault attack. A hardware fault attack example is given here for a two-prime RSA. s'_p is denoted as the faulty value of s_p, induced by some errors due to the interference of the attacker. Then, a faulty signature s' will be computed based on the fault-free values of s_q and s'_p. After the attacker intercepts the faulty signature s', he will be able to factor the modulus of the two-prime RSA system by computing $q = \gcd((s'^e -$

$m) \bmod n, n)$ and $p = n/q$ [1].

Various methods of fault injections and their effects are reviewed in [8]. These techniques include: variations in supply voltage and external clocks, temperature, white light, laser, X-ray and ion beam, which can be categorized into four models of attacks [5]. We list the summary of these four models from [27].

- Fault Model #1: The attacker can cause a fault in a single bit. The attacker has full control over the timing and location of the fault.
- Fault Model #2: The attacker can cause a fault in a single byte. The attacker has full control over the timing, but may have only partial control over the location of the fault (e.g., which byte or bit is affected), and cannot predict the new faulty value that is introduced.
- Fault Model #3: The attacker can cause a fault in a single byte of a variable. The attacker has only partial control over the timing and location of the fault, and cannot predict the new faulty value.
- Fault Model #4: The attacker can cause a fault in a single variable. The attacker has only partial control over the timing and no control over the location; the targeted variable will be replaced by an entirely new random value not known to the attacker.

Note that these models are listed in order of decreasing attacker power, so a scheme that is secure against one fault model will be secure against all higher-numbered models as well. Also, the models 3 and 4 are the most common practical attacks.

2.3 Countermeasure Study

There has been extensive study in the literature to search for some way to secure multi-prime RSA against hardware fault attacks [2, 3, 5, 6, 8, 12, 19, 25, 27, 29, 30, 31, 32, 33, 34, 35]. Countermeasures without checking procedure are desirable for RSA cryptosystems [33]. Typical non-checking methods are the CRT-1 and CRT-2 protocols proposed by Yen et. al. [33] and the BOS algorithm [5] based on a notion of infective computation. These techniques revise the calculation expression of the signature so that the public modulus will not be factored even if faulty signatures are sent out of the system. It has been proven that the complexity to factor the modulus is $O(n)$ for both CRT-1 and CRT-2 protocols when the attacker intercepts a faulty signature. CRT-1 requires a serial operation, which operates in a low computational speed. In CRT-2, the calculations can be performed in parallel. Therefore, CRT-2 provides higher speed than CRT-1. The computation steps in CRT-2 for a two-prime RSA are as follows.

In a two-prime RSA, $j = 2$ and the modulus n is a large number with two prime factors i_1 and i_2. These two prime numbers are often denoted as $i_1 = p$ and $i_2 = q$ such that $n = p \cdot q$. Also, we have

$$\begin{aligned} k_p &= \lfloor m/p \rfloor \\ k_q &= \lfloor m/q \rfloor. \end{aligned}$$

Definition 1. *t is chosen to be a small integer and a key pair (e_t, d_t) is selected, where*

$$\begin{aligned} d_t &= d - t \\ e_t &\equiv d_t^{-1} (\bmod \lambda(n)). \end{aligned}$$

The CRT-2 protocol calculates the signature s as follows:

Step 1. Compute S_{pt} and S_{qt} as follows:

$$\begin{aligned} S_{pt} &= (m \bmod p)^{d_t mod(p-1)} \bmod p \\ S_{qt} &= (m \bmod q)^{d_t mod(q-1)} \bmod q. \end{aligned}$$

Step 2. The signature s is calculated by including a product term to the CRT expression as

$$\begin{aligned} s &= (CRT(s_{pt}, s_{qt}) \cdot \tilde{m}^t) \bmod n \quad \text{where} \qquad (2) \\ \tilde{m} &= \lfloor \frac{(s_{pt}^{e_t} \bmod p + k_p \cdot p + s_{qt}^{e_t} \bmod q + k_q \cdot q)}{2} \rfloor. \end{aligned}$$

The Equation (2) will provide the correct result when there is no hardware fault attack. When there is an attack, the result will be randomized to prevent the attacker from doing the factorization. The CRT-2 algorithm given above is for a two-prime case and can be easily extended to the general multi-prime case. However, it is discovered in [30] that if the attacker will inject an error in k_p or k_q, the system will be broken. The attack is as follows.

Provided that k_p, s_p and s_q are correct and k_q becomes incorrect, we get $s' = (m^d + R \times q) \bmod n$, where R is a random number due to the incorrect k_q. This will lead to

$$q = \gcd((s'^e - m), n). \qquad (3)$$

Similarly,a fault on k_p leads to

$$p = \gcd((s'^e - m), n). \qquad (4)$$

In order to improve the CRT-2 in view of the attack of [30], the BOS algorithm [5] was proposed by using two parallel structures and a more involved algorithm. However, it is then later discovered that another attack in [27] can break the BOS algorithm.

3 Proposed Countermeasure Method

In this section, we propose a new countermeasure method without using k_p and k_q variables. Then, based on the properties of the FPGA implementation, we carry out a cryptanalysis and comparison study.

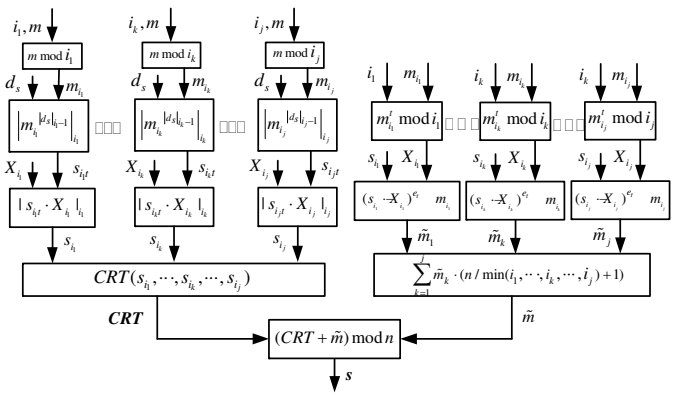

Figure 2: The block diagram of the proposed multi-prime RSA

3.1 Proposed Countermeasure Method

Definition 2. *We use a new key pair of* (e_t, d_t)*, where* t *is an integer smaller than* d*, and*

$$\begin{aligned} d_t &= d + t \\ e_t &\equiv d_t^{-1} (\mathrm{mod} \lambda(n)) \\ d_s &= d - t. \end{aligned}$$

Proposed countermeasure: Based on Definition 2, the signature of the proposed j-prime RSA is calculated by the following steps. Let $k = 1, \cdots, j$, we have

- Step 1: Compute $m_{i_k} = m \bmod i_k$.
- Step 2: Compute $X_{i_k} = m_{i_k}^t \bmod i_k$ and $s_{i_k t} = m_{i_k}^{d_s \bmod (i_k - 1)} \bmod i_k$.
- Step 3: Compute $s_{i_k} = (s_{i_k t} \cdot X_{i_k}) \bmod i_k$, then the signature s is computed by adding a perturbing summing term to the standard CRT operation as $s = (CRT(s_{i_1}, \cdots, s_{i_j}) + \tilde{m}) \bmod n$, where $\tilde{m} = (\sum_{k=1}^{j} ((s_{i_k} \cdot X_{i_k})^{e_t} \bmod i_k - m_{i_k})) \cdot (\frac{n}{min(i_1, \cdots, i_j)} + 1)$.

Based on the above algorithm, the block diagram of the proposed multi-prime RSA is shown in Figure 2. This system will give the correct signature when there is no hardware fault attack as explained next.

Proposition 1. *When there is no hardware fault attack,* $s = (CRT(s_{i_1}, \cdots, s_{i_j}) + \tilde{m}) \bmod n = CRT(s_{i_1}, ..., s_{i_j})$.

Proof. If there is no hardware fault attack and let $k = 1, \cdots, j$:

$$\begin{aligned} s_{i_k t} &= m_{i_k}^{d_s \bmod (i_k - 1)} \bmod i_k \\ &= (m \bmod i_k)^{d_s \bmod (i_k - 1)} mod i_k = m^{d_s} \bmod i_k. \\ s_{i_k} &= s_{i_k t} \cdot X_{i_k} \\ &= m^{d_s} \bmod i_k \cdot m_{i_k}^t \bmod i_k \\ &= (m^{d_s} \cdot m^t) \bmod i_k = m^d \bmod i_k. \\ &= (s_{i_k} \cdot X_{i_k})^{e_t} \bmod i_k - m_{i_k} \\ &= (m^d \bmod i_k \cdot m^t \bmod i_k)^{e_t} \bmod i_k - m \bmod i_k \\ &= (m^{d_t} \bmod i_k)^{e_t} \bmod i_k - m \bmod i_k \\ &= m^{1 + k\lambda(n)} \bmod i_k - m \bmod i_k = 0. \end{aligned}$$

Thus,

$$\begin{aligned} \tilde{m} &= (\sum_{k=1}^{j} ((s_{i_k} \cdot X_{i_k})^{e_t} \bmod i_k - m_{i_k})) \\ &\quad \cdot (\frac{n}{min(i_1, \cdots, i_j)} + 1) \\ &= \sum_{k=1}^{j} 0 \cdot (\frac{n}{min(i_1, \cdots, i_j)} + 1) = 0. \end{aligned}$$

Therefore, $s = (CRT(s_{i_1}, \cdots, s_{i_j}) + \tilde{m}) \bmod n = CRT(s_{i_1}, \cdots, s_{i_j})$,which is the same as the correct result Equation (1). □

When there is hardware fault attack, the proposed countermeasure can ensure security. In the following paragraphs, we discuss two typical attacks. The attacker can keep s_{i_k} correct and insert byte error or random error to make $\tilde{m}$ faulty. Based on a faulty $\tilde{m}$, the complexity for the attacker to obtain any factor of the proposed multi-prime RSA is $O(n)$.

Proposition 2. *Given a j-prime RSA, a faulty signature* s'_k *is obtained by an attacker from a faulty value* $\tilde{m'}$*. The complexity of obtaining one factor* i_k *is* $O(n)$*, by using* $i_k = n/gcd((((s'_k - \Delta) \bmod n)^e - m) \bmod n, n)$*, where* Δ *and* k *are integers,* $\Delta \in [0, n)$*, and* $k \in [1, j]$.

Proof. When calculating the signature, we assume all the other parts are correct except $\tilde{m}$. The faulty value of $\tilde{m'}$ will be a random number. This means a random value is added to the signature. If the attacker tries to use the wrong signature to factor n, he has to eliminate the value of $\tilde{m'}$, by using $i_k = gcd((((s'_k - \Delta) \bmod n)^e - m) \bmod n, n)$ where Δ is an integer and $\Delta \in [0, n)$. The only possible way for him to do so is to guess Δ in the range of n. Thus, the complexity for the attacker to obtain i_k from this faulty signature s'_k is $O(n)$. □

The attacker can also inject byte error or random error to make s_{i_k} faulty. Based on a faulty s_{i_k}, the complexity for the attacker to obtain any factor of the proposed multi-prime RSA is still $O(n)$. We will need the following Lemma to derive Proposition 3.

Lemma 1. *Given that* p *is a prime number,* a, b, c *are three integers smaller than* p*, and* $b \neq c$*, we have* $ab \mathrm{mod} p \neq ac \mathrm{mod} p$.

Proposition 3. *Given a j-prime RSA, a faulty signature* s'_k *is obtained by an attacker from a faulty value* s'_{i_k}*. The complexity of obtaining one factor* i_k *is* $O(n)$*, by using* $i_k = n/gcd((((s'_k - \Delta) \bmod n)^e - m) \bmod n, n)$*, where* Δ *and* k *are integers,* $\Delta \in [0, n)$*, and* $k \in [1, j]$.

Proof. Based on the above Lemma 1 and Lemma 4 in [33], each erroneous value s'_{i_k} will produce a wrong result of $\tilde{m}$. Totally, there are $p-1$ possible values of s'_{i_k}, which will produce $p-1$ possible erroneous values of $\tilde{m}$ in the range of $[0, n)$. When the signature s is calculated using (5) with the wrong result of $\tilde{m}$, a value in the range of $[0, n)$ will be added as an interfering factor.

When the attacker uses s'_k to factor n, he has to eliminate the influence of $\tilde{m}$ from s'_k, by using $i_k = gcd((((s'_k - \Delta) \bmod n)^e - m) \bmod n, n)$ where Δ is an integer and $\Delta \in [0, n)$. The only possible way for him to do so is to guess the value of Δ. Thus, the complexity for the attacker to obtain i_k from the faulty signature s'_k is $O(n)$. □

It is noted that the complexity of obtaining one factor from a faulty signature in CRT-2 protocol is also proven to be $O(n)$ in [33]. Thus, based on Propositions 2 and 3, the proposed method provides a similar immunity against the two typical hardware fault attacks.

3.2 Comparison with CRT-2

The proposed method can be considered as a revised version of CRT-2. It removes k_p and k_q variables in the CRT-2 so that it can survive the attack of [30] that broke CRT-2. We use the following two-prime case with the modulus $\{p, q\}$ for illustration. If the attacker keeps the rest of the system correct and inserts errors in $\tilde{m}_{i_k} = p$, the result $\tilde{m}$ will not be '0' but a $\tilde{m}' = R \cdot (q+1)$ where R is a random number. Then, we got

$s' = (CRT(s_{i_1}, \cdots, s_{i_j}) - R \cdot (q+1)) \bmod n$.

Thus, $gcd((s'^e - m, n)$ (see Equations (3) and (4)) will not generate q to factor n.

The proposed method not only can survive the attack of [30] but also is efficient for the FPGA implementation. We compare the implementation aspects of the proposed method with those of the CRT-2 protocol in Table 1. It is clear that the operations of the proposed multi-prime RSA use smaller numbers than those of the CRT-2 protocol. Moreover, extra dividers are needed to perform the division computations of K_{i_k} and $\tilde{m}/j$ in CRT-2 protocol, while no divider is needed in the proposed multi-prime RSA.

3.3 FPGA

In order to further discuss the effectiveness of the proposed countermeasure, we focus on the proposed countermeasure from the perspective of FPGA implementation of multi-prime RSA. To ensure security in real practice, one must consider the properties of different implementations of cryptosystems. FPGA implementation is very useful for RSA cryptosystems [4, 7, 10, 11, 15, 16, 18, 23, 28] and can have some advantages against various hardware fault attacks. Introduced in 1985 by Xilinx, FPGA is a high capacity programmable logic device consisting of an array of programmable logic cells surrounded by programmable interconnects. Its logic cells and interconnects can be programmed by end-user to implement specific circuitry. It has many applications in prototyping, FPGA-based computers, on-site hardware reconfiguration, DSP, logic emulation, network components, and cryptography. One advantage of using FPGA for multi-prime RSA is that the logic cells and interconnects used for one variable can be easily scattered. This will somehow limit the attacker's control of the timing and location of the variable. In the logic synthesis step of FPGA design, one can use synthesis file (for example, UCF file in Xilinx FPGA design) to let the logic cells and interconnects spread all over the whole FPGA chip. Different bits of one variable will not be in adjacent locations but scattered in different locations. Therefore, for the FPGA-based multi-prime RSA, the hardware fault attacks will be similar to the model 3 and model 4 [5] with the limitation that the attack can not locate the data bus. In the microprocessor and data bus structure, the attacker can find the data bus location and change the value of a variable to some parts of the design, but keep the original value of the variable to other parts of the design. However, due to the programmability, and place and routing specialty of the interconnect network in the FPGA, the data bus or interconnects of the variable is scattered all over the chip and difficult to find its direction. Therefore, the attack can inject error in the logic cell or interconnect used for one variable and this change will goes to every part of the design.

The importance of using FPGA to design the proposed countermeasure is that FPGA utilizes the advantages of the proposed structure, in which the CRT part and $\tilde{m}$ part are scrambled together. As shown in Figure 2, several variables are shared between the CRT part and $\tilde{m}$ part such as s_{i_k}, i_k, X_{i_k}, and m_{i_k}. We use the following example to illustrate the advantage. The attacker can inject errors in the common variable such as m_{i_k} to make both s_{i_k} and $\tilde{m}$ faulty. Due to the scattered interconnects of FPGA, the attacker cannot find the specific location of the data bus of m_{i_k} for s_{i_k} to inject errors without changing m_{i_k} in the $\tilde{m}$ part. In another word, the faulty $\tilde{m}$ and faulty s_{i_k} will be based on the same wrong input m_{i_k}. Thus, it is similar to the input being another number and the attacker can not find the different effects of the faulty $\tilde{m}$ and the faulty s_{i_k} and can not use this information to break the system.

3.4 Various Attacks

In [27], in order to break the BOS algorithm, the attacker needs to inject errors when the system reads m and keep the correct value of m in the memory. This assumption is correct for the structure of microprocessor and data bus. However, based on the discussion of Section 3.3, the attacker can not have this ability for the FPGA implementation.

In the literature, the countermeasure study of multi-prime RSA is generally based on the microprocessor and data bus structure. This leads to several attacks to eas-

Table 1: Performance comparison

	Division	Base	Modular exponentiation	Modular exponentiation
CRT-2 protocol [20]	$K_k = \lfloor m/i_k \rfloor$ and $\tilde{m}/j$	m	$m^{d_t} \bmod i_k$ $(m \in [1, n))$	$\tilde{m}^t \bmod n$ $(m \in [1, n))$
The proposed multi-prime RSA	-	$m_{i_k} = m \bmod i_k$	$m_{i_k}^{d_t} \bmod i_k$ $(m_{i_k} \in [1, i_k))$	$m_{i_k}^{t} \bmod i_k$ $(m_{i_k} \in [1, i_k))$

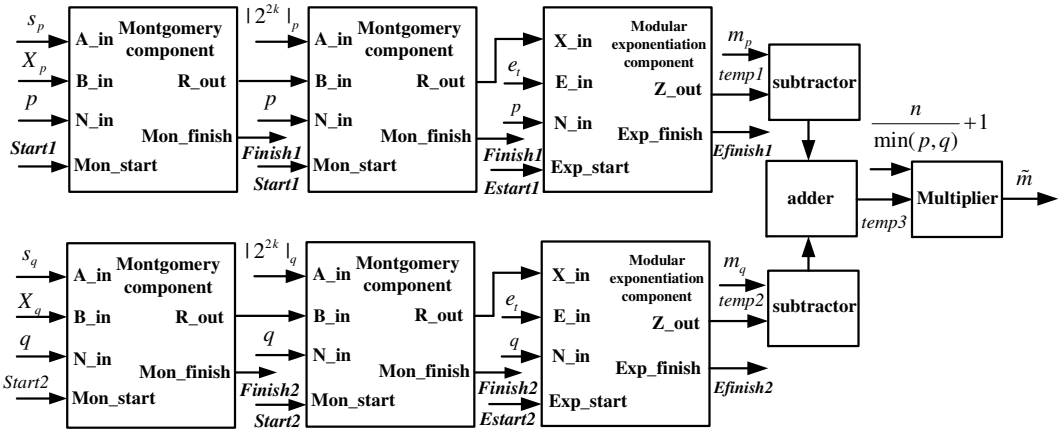

Figure 3: The implementation of $\tilde{m}$ for the proposed two-prime immune RSA

ily break these countermeasure methods. We propose to study the countermeasure method for FPGA-based multi-prime RSA. The FPGA structure limits the control of the attacker of the location and timing of the data bus. Therefore, FPGA might be an efficient solution for secure cryptosystem designs.

Also, it is worthy to notice that the BOS algorithm [5] might also be effective for the FPGA implementation, since the attack to break BOS algorithm in [27] is not valid for FPGA implementation. However, the BOS algorithm uses a more complicated algorithm than the proposed method. Therefore, the proposed method is more practical for the actual RSA cryptosystem designs.

4 FPGA Implementation And Testing

In this section, we carry out the FPGA implementation and testing for two-prime and three- prime cases. The implementation diagrams are shown in Figures 3, 4, 5, and 6.

4.1 FPGA Implementation

The FPGA design approach includes the following steps: design entry, logic verification (simulation), synthesis, implementation design (translating, mapping, placing and routing). The software used is Xilinx Integrated Synthesis Environment (ISE) 6.1i version. Its Graphic User Interface (GUI) consists of push-button flows where a design can be specified either in Hardware Description Language (HDL) form or in schematic form. It also includes all integrated tools for all the above steps and the interface with the third party software. In our design, the VHDL format is adopted as the design entry. The Modelsim XE 5.7 version software is used for Boolean logic function verification. The synthesis step of the VHDL code is accomplished by the Xilinx Synthesis Technology (XST) tool. In the synthesis step, the design is scattered all over the chip to reduce the attacker's ability. Then, the implementation design step is executed to compile the synthesized design with the translating, mapping and placing & routing tools available from Xilinx. This is the step to generate the layout of the design on the selected chip. The ISE software also provides an integrated tool "core generator", which includes some Intellectual Property (IP) cores. The core generator tool generates the needed dividers in the implementations. The xcv2000-bg560 device is chosen for the implementation.

It is worthy to notice that in [4, 7], multi-prime RSA systems with immunity have been implemented using FPGA. The design in [7] implemented a system consisting of the CRT-2 and other immunity algorithms for a two-prime case. Therefore, we use the same RSA parameters such as the modulus n, the secret key d and the message

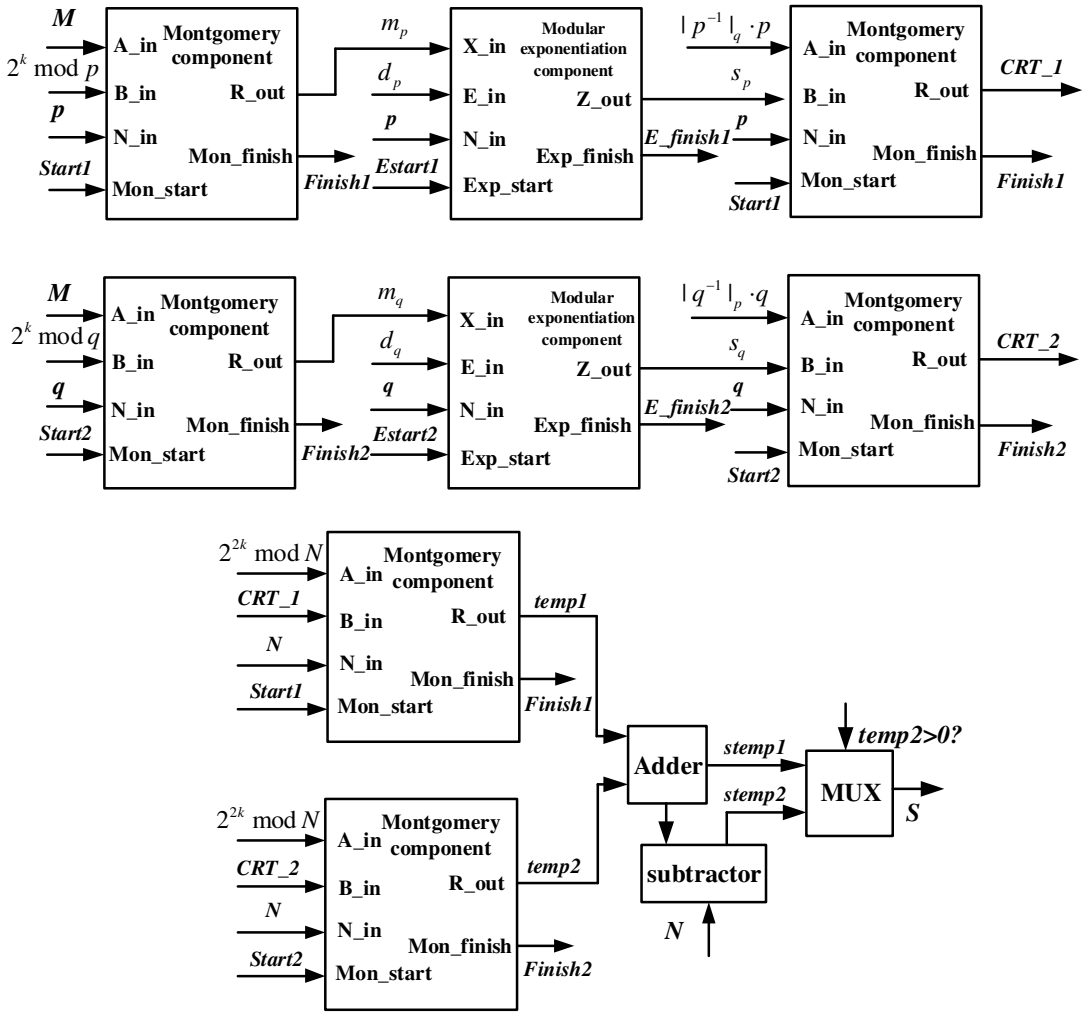

Figure 4: The implementation of the CRT part of the proposed two-prime immune RSA

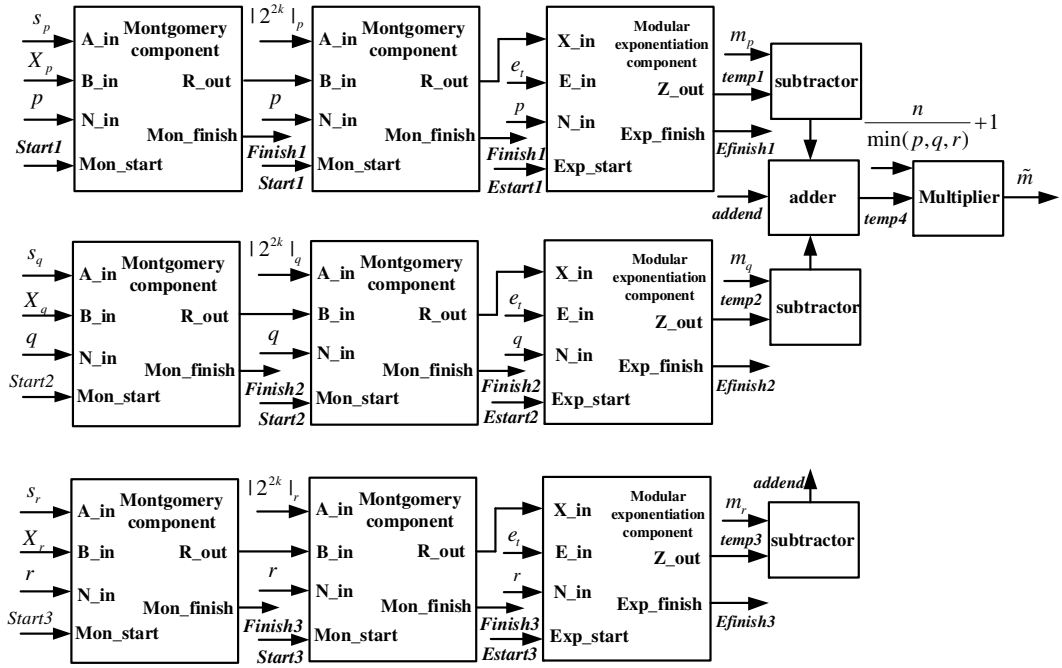

Figure 5: The implementation of $\tilde{m}$ for the proposed three-prime immune RSA

m as in [7]. The key length is also 512-bit. In our design, in order to save hardware resources, modular exponentiation calculations [17, 21, 26] for s_p and s_q are reused for the calculation of $\tilde{m}$.

The implementation results of the proposed two-prime and three-prime RSAs, including the configurable logic block (CLB) usage, the look up table (LUT) usage, the equivalent gates and the data rate, are shown in Table 2. For the purpose of comparison, the implementation data of the standard two-prime and three-prime, CRT-2-based two-prime and three-prime RSA cryptosystems, and the two-prime design in [7] are also shown in Table 2.

It is seen from Table 2 that for two-prime and three-prime RSAs, the proposed method requires only 66% and 75% of hardware resources respectively (the percentage of equivalent gate count) while providing higher operating speed, compared with the CRT-2 protocol or the one in [7]. The reason is as follows. For the proposed multi-prime RSA, the operations of the summing term $\tilde{m}$ to increase the immunity are performed mainly by reusing the hardware resources for the previous modular exponentiation and modular multiplication operations. This assures that the summing term does not require a lot of resources usage.

It is also noted that the proposed design uses a scattered structure, which can be more efficient in view of hardware fault attacks. However, based on CRT-2, the design [7] still used a microprocessor and data bus structure inside the FPGA. Therefore, it might not survive the attacks proposed in [27] and [30].

4.2 Testing

In order to verify the effectiveness of the proposed countermeasure, we carried out a number of testing for the three-prime design in view of different hardware fault attacks. The testing setup is shown in Figure 7. We first carried out the testing of the attacks such as variations in supply voltage and external clocks, and temperature. These attacks can inject random errors for all the components of the FPGA chip. The proposed system will not provide useful information for the attacker to break the system. Then, we carried out the testing for the laser fault attack for specific locations of the FPGA chip. The laser fault injection equipment is similar to the one used in [8] and shown in Figure 8. The attack can inject byte or random errors in variables of the system. But due to the scattering of the FPGA logic cell and interconnects, the attack can not locate specific data bus. It has been verified that the proposed systems can handle such an attack. Due to the limitation of our lab, other attacks such as white light, X-ray and ion beam have not been tested.

5 Conclusion

In this paper, a study of the immunity algorithm, structure and implementation of the Chinese remainder theorem-based multi-prime RSA has been carried out. Based on the FPGA implementation concepts, a novel countermeasure method is proposed, which can survive the attacks [27, 30] that broke the previous methods [5, 33]. By using a simple operation and small wordlength

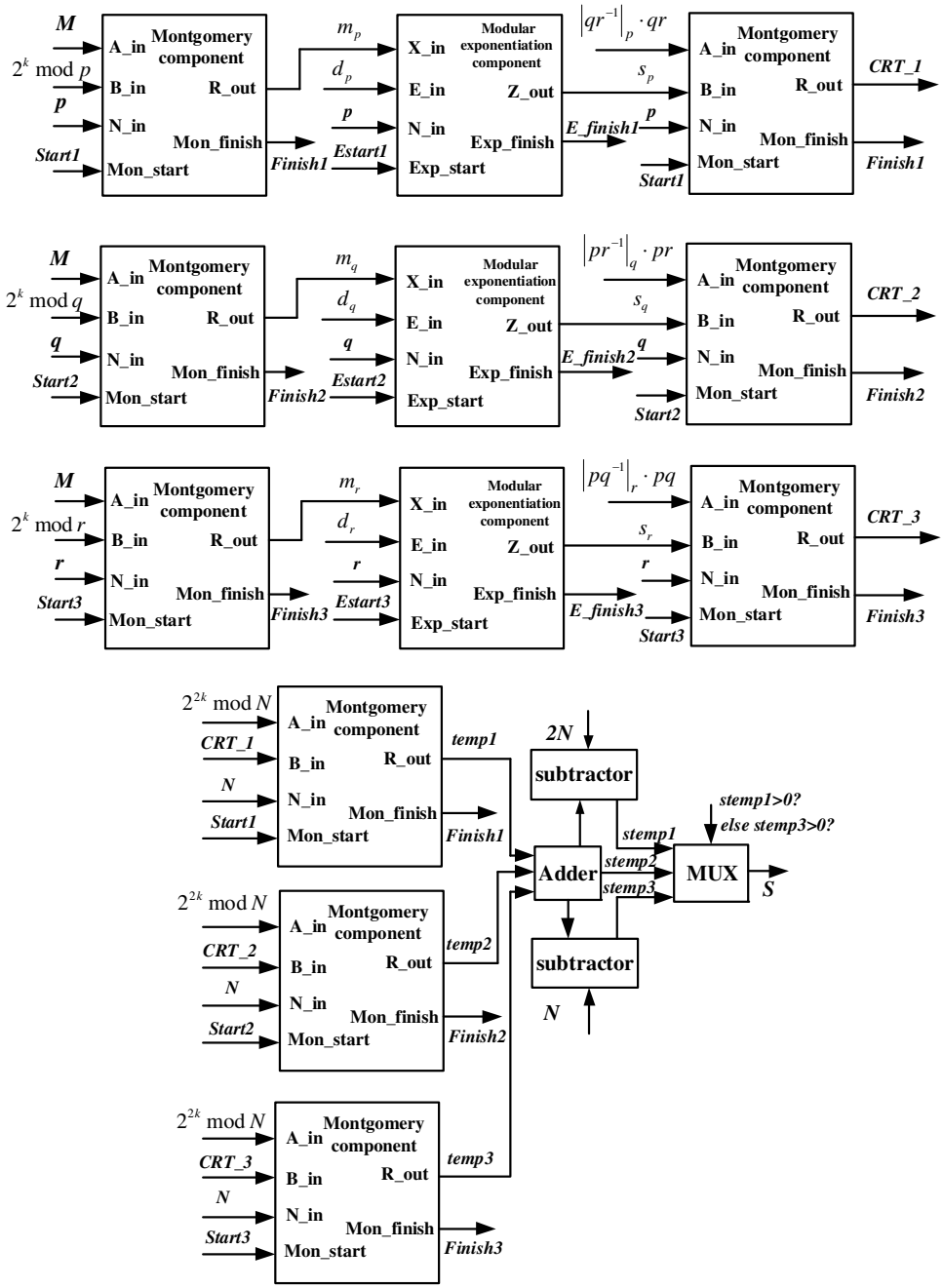

Figure 6: The implementation of the CRT part of the proposed three-prime immune RSA

Table 2: FPGA implementation results using xcv1000-5bg560 device

	CLB usage	LUT usage	Equivalent gates	Data rate
Two-prime RSA based on standard CRT	1226	4775	46324	1.13Mbits.s
Two-prime RSA based on CRT-2 protocol [33]	1997	6577	85229	0.59Mbits/s
Two-prime RSA based on method in [7]	2788	8185	>90,000	0.03Mbits/s
Proposed two-prime RSA	**1,431**	**5,615**	**55,913**	**0.68Mbits/s**
Three-prime RSA based on standard CRT	1,759	6,939	68,144	1.74 Mbits/s
Three-prime RSA based on CRT-2 protocol [33]	2,646	9,121	109,756	0.75 Mbits/s
Proposed three-prime RSA	**2,130**	**8,252**	**82,233**	**0.98 Mbits/s**

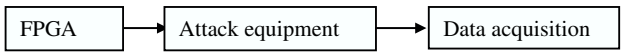

Figure 7: The block diagram of the testing setup

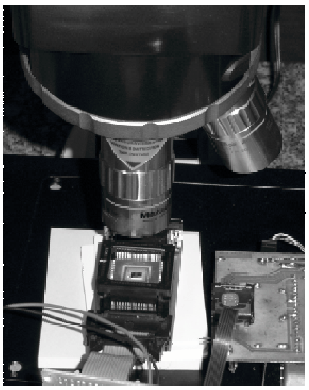

Figure 8: Laser fault injection equipment

parameters, the proposed method is also efficient in terms of hardware resources and speed. For a two-prime RSA and a three-prime RSA, the proposed method requires only 66% and 75% hardware resources respectively while providing higher operating speed, compared with the existing method. In order to prove the functionality of the proposed countermeasure, the FPGA implementation and testing in attacking environment are carried out for several two-prime and three-prime design examples. Utilizing the properties of FPGA implementations in the design of multi-prime RSA cryptosystems can limit the ability of the attacker to control the timing and location of the cryptosystem. This should be useful in the design of many other secure cryptosystems. It is expected that the proposed techniques will have a positive impact on the hardware implementation of cryptosystems and will be used in the applications of computer networking, telecommunications, and smart card.

References

[1] R. J. Anderson and M. G. Kuhn, "Tamper resistance—a cautionary note," in *The 2nd USENIX Workshop on Electronic Commerce*, pp. 1-11, 1996.

[2] R. J. Anderson and M. G. Kuhn, "Low cost attacks on tamper resistant devices," in *Proceedings of the 5th International Workshop on Security Protocols*, pp. 125-136, 1997.

[3] C. Aumüller, P. Bier, W. Fischer, P. Hofreiter, and J.-P. Seifert, "Fault attacks on RSA with CRT: Concrete results and practical countermeasures," in *the 4th International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 260-275, 2002.

[4] J. C. Bajard and l. Imbert, "A full RNS implementation of RSA," *IEEE Transactions on Computers*, vol. 53, no. 6, pp. 769-774, Jun. 2004.

[5] J. Blomer, M. Otto, J. P. Seifert, "A new CRT-RSA algorithm secure against bellcore attacks," in *10th ACM conference on Computer and Communications Security*, pp. 311-320, 2003.

[6] D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the importance of checking cryptographic protocols for fault," in *Eurocrypt'97*, LNCS 1233, pp. 37-51, Springer-Verlag, 1997.

[7] M. Ciet, M. Neve, E. Peeters, and J. J. Quisquater, "Parallel FPGA implementation of RSA with residue number systems - can side-channel threats be avoided?," in *Proceedings of the 46th IEEE Midwest Symposium on Circuits and Systems*, Dec. 2003.

[8] H. B.-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan, "The sorcerer's apprentice guide to fault attacks," in *Workshop on Fault Detection and Tolerance in Cryptography*, pp. 370-383, June 2004.

[9] E. English and S. Hamilton, "Network security under siege: The timing attack," *IEEE Transaction Computer*, vol. 29, pp. 95-97, 1996.

[10] J. Grossschadl, "The chinese remainder theorem and its application in a high-speed RSA crypto chip," in *The 16th Annual Conference on Computer Security Applications*, pp. 384-393, Dec. 2000.

[11] M. K. Hani, T. S. Lin, and S. H. Nasir, "FPGA implementation of RSA public-key cryptographic coprocessor," in *Proceedings on TENCON'00*, vol. 3, pp. 6-11, Sep. 2000.

[12] M. Joye, A. K. Lenstra, and J.-J. Quisquater, "Chinese remaindering based cryptosystems in the presence of faults," *Journal of Cryptology*, vol. 12, no. 4, pp. 241-245, 1999.

[13] P. Kocher, "Timing attacks on implementations of Die-Hellman, RSA, DSS, and other systems," in *Proceedings of Crypto'96*, pp. 104-113, Springer-Verlag, 1996.

[14] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proceedings of Crypto'99*, pp. 388-397, Aug. 1999.

[15] C. K. Koc, *High-speed RSA Implementations*, Technical notes TR 201, RSA Security Inc., Nov. 1994.

[16] C. K. Koc, *RSA hardware implementation*, Technical notes TR 801, RSA Security Inc., Aug. 1995.

[17] P. Kornerup, "A systolic, linear-array multiplier for a class of right-shift algorithms," *IEEE Transactions on Computer Arithmetic*, vol. 43, pp. 892-898, Aug. 1994.

[18] Krishnamurthy, Y. Tang, C. Xu, and Y. Wang, "An efficient implementation of multi-prime RSA on DSP processor," in *IEEE International Conference on Acoustics, Speech, & Signal Processing*, vol. 2, pp. 413-416, Apr. 2003.

[19] A. K. Lenstra, "Memo on RSA signature generation in the presence of faults," pp. 1-4, Sep. 1996.

[20] T. S. Messerges, *Power Analysis Attack Countermeasures and their Weaknesses*, Security Technology Research Laboratory, 2000.

[21] P. L. Montgomery, "Modular multiplication without trial division," *Mathematics of Computation*, vol. 44, pp. 519-521, 1985.

[22] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, pp. 120-126, Feb. 1978.

[23] RSA Laboratories, *PKCS #1 v2.0 Amendment 1: Multi-Prime RSA*, Jul. 2000. (ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-0a1.pdf)

[24] W. Schindler, "A timing attack against RSA with the chinese remainder theorem," in *Proceedings of Cryptographic Hardware and Embedded Systems*, pp. 109-124, 2000.

[25] A. Shamir, "How to check modular exponentiation," in *Proceedings of Eurocrypt'97*, pp. 123, May 1997.

[26] C. Su, S. Hwang, P. Chen, and C. Wu, "An improved montgomery's algorithm for high-speed RSA public-key cryptosystem," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 7, pp 280-284, June 1999.

[27] D. Wagner, "Cryptanalysis of a provably secure CRT-RSA algorithm," in *Proceedings of the 11th ACM conference on Computer and communications security*, pp. 92-97, 2004.

[28] C. H. Wu, J. H. Hong, and C. W. Wu, "RSA cryptosystem design based on the chinese remainder theorem," in *Proceedings of the ASP-DAC'01*, pp. 391-95, 2001.

[29] S. M. Yen and M. Joye, "Checking before output may not be enough against fault-based cryptanalysis," *IEEE Transactions on Computers*, vol.49, no. 9, pp. 967-970, Sep. 2000.

[30] S. M. Yen and D. Kim, "Cryptanalysis of two protocols for RSA with CRT based on fault infection," in *Proceedings of the Workshop on Fault Diagnosis and Tolerance in Cryptography*, pp. 381-385, June 2004.

[31] S. M. Yen, S. Kim, S. Lim, and S. J. Moon, "A countermeasure against one physical cryptanalysis may benefit another attack," in *Proceedings of the 4th International Conference Seoul on Information Security and Cryptology*, pp. 414-427, Dec. 2001.

[32] S. M. Yen, S. Kim, S. Lim, and S. J. Moon, "RSA speedup with residue number system immune against hardware fault cryptanalysis," in *Proceedings of the 4th International Conference Seoul on Information Security and Cryptology*, pp. 397-413, Dec. 2001.

[33] S. M. Yen, S. Kim, S. Lim, and S. J. Moon, "RSA Speedup with chinese remainder theorem immune against hardware fault cryptanalysis," *IEEE Transactions on computers*, vol. 52, pp. 461-472, Apr. 2003.

[34] S. M. Yen, S. J. Moon, and J. C. Ha, "Hardware fault attack on RSA with CRT Revisited," in *Proceedings of the ICISC'02*, LNCS 2587, pp. 374-388, Springer-Verlag, 2003.

[35] S. M. Yen, S. J. Moon, and J. C. Ha, "Permanent fault attack on RSA with CRT," in *Proceedings of the ACISP'03*, LNCS 2727, pp. 285-296, Springer-Verlag, 2003.

Zine-Eddine Abid received the M.Sc. and Ph.D. degrees from the University of Minnesota, Minneapolis, USA, in 1987 and 1991 respectively. He received the B.Sc. degree in 1983 from University of Setif, Algeria. He was a Research Associate and Lecturer at the University of Minnesota during 1991/1992. He worked at National Research Council (NRC), Ottawa, on the fabrication, design, and characterization of high frequency III-V Heterostructure Bipolar Transistors (HBTs). He joined Nortel networks in 1995, to work on GaAs HBTs processing for high frequency integrated circuits. He was an Assistant Professor with the Department of Electrical Engineering, King Saud University, Riyadh, KSA. He has been a faculty member at the Department of Electrical and Computer Engineering, University of Western Ontario, Canada, since September 2002, working on CRT-based RSA Cryptosystems and CMOS VLSI Circuits, targeting low power, high speed and defect tolerant designs. He is the author and co-author of more than 40 conference and journal papers.

Wei Wang received his B. Sc. degree in 1992 from the Beijing University of Aeronautics, China, and his Ph. D degree in 2002 from Concordia University, Montreal, QC, Canada, both in Electrical and Computer Engineering. From 2000 to 2002, he served as an ASIC and FPGA design engineer in EMS technologies, Montreal, QC, Canada. From 2002 to 2004, he was a faculty member in the Department of Electrical and Computer Engineering, the University of Western Ontario, London, ON, Canada. From 2004, he has been a professor in the Department of Electrical and Computer Engineering, Indiana University-Purdue University at Indianapolis (IUPUI).

His main research interests are VLSI, nanoelectronics, DSP, cryptography, digital design, ASIC and FPGA design, and computer arithmetic. He has over 60 journal and conference publications in these areas. He has published over 80 journal and conference papers. He has one US patent and another patent pending. He received Canadian Foundation of Innovation Award (nanoelectronics) in 2004, Indiana Purdue University Research Initiative Award (nanoelectronics) in 2005 and Best Paper Award from IEEE CCECE conference in 2005.