8

Scan-Based Side-Channel Attack on the RSA Cryptosystem

Ryuta Nara, Masao Yanagisawa and Nozomu Togawa
Waseda University
Japan

1. Introduction

Individual authentication increases in importance as network technology advances. IC passport, SIM card and ID card used in entering and leaving management systems are dependent on a cryptography circuit for keeping their security. LSI chips used there usually include cryptography circuits and encrypt/decrypt important data such as ID numbers and electronic money information. However, there is a threat that a secret key may be retrieved from the cryptography LSI chip. Recently, side-channel attacks against a cryptosystem LSI has been reported (Boneh et al., 1997; Brier et al., 2004; Kocher, 1996; Kocher el al., 1999; Schramm el al., 2003). For example, scan-based side-channel attacks which retrieve secret keys in a cryptography LSI have attracted attention over the five years. A scan path is one of the most important testing techniques, where registers are connected in serial so that they can be controlled and observed directly from outside the LSI. Test efficiency can be increased significantly. On the other hand, one can have register data easily by using a scan path, which implies that one can retrieve a secret key in a cryptography LSI. This is a *scan-based side-channel attack.*

One of the difficulties in the scan-based side-channel attack is how to retrieve a secret key from obtained scanned data from a cryptosystem LSI. In a scan path, registers inside a circuit have to be connected so that its interconnection length will be shortened to satisfy timing constraints. This means that no one but a scan-path designer knows correspondence between registers and scanned data. To succeed a scan-based side-channel attack against a cryptography LSI, an attacker needs to retrieve secret keys from the scanned data almost "randomly" connected.

Symmetric-key cryptosystems such as DES and AES are very popular and widely used. They make use of the same secret key in encryption and decryption. However, it may be difficult to securely share the same secret key, such as in communicating on the Internet. Public-key cryptosystems, on the other hand, make use of different keys to encrypt and decrypt. One of the most popular public-key cryptography algorithms is RSA (Rivest et al., 1978), which is used by many secure technologies such as secure key agreement and digital signature.

Yang et al. first showed a scan-based side-channel attack against DES in 2004 and retrieved a secret key in DES (Yang et al., 2004). They also proposed a scan-based side-channel attack against AES in 2006 (Yang et al., 2006). Nara et al. proposed an improved scan-based side-channel attack method against AES in 2009 (Nara et al., 2009). A scan-based side-channel

attack against elliptic curve cryptography (Koblitz, 1987; Miller, 1986) was proposed by Nara et al. (Nara et al., 2011). On the other hand, any scan-based side-channel attacks against RSA have not been proposed yet in spite of the fact that RSA is a de-facto standard for a public-key cryptosystem. Since public-key cryptosystems have complicated algorithm compared with that of symmetric-key cryptosystems such as DES and AES, we cannot apply the scan-based side-channel attacks against symmetric-key cryptosystems to an RSA circuit. An elliptic curve cryptography algorithm is completely different from an RSA algorithm although they both are public-key algorithms. We cannot apply the scan-based side-channel attacks against elliptic curve cryptosystem to RSA, either.

In this paper, we propose a scan-based side-channel attack against an RSA circuit, which is almost independent of a scan-path structure. The proposed method is based on detecting intermediate values calculated in an RSA circuit. We focus on a 1-bit time-sequence which is specific to some intermediate value. We call it a *scan signature* because its value shows their existence in the scanned data obtained from an RSA circuit. By checking whether a scan signature is included in the scanned data or not, we can retrieve a secret key in the target RSA circuit even if we do not know a scan path structure, as long as a scan path is implemented on an RSA circuit and it includes at least 1-bit of each intermediate value.

The purpose of our proposed method is, not to make secure scan architecture ineffective but to retrieve a secret key using scanned data in an RSA circuit with as few limitations as possible. In fact, our scan-based side-channel attack method without any modification might not work against RSA circuits using some secure scan architecture. Several secure scan architectures without consideration of our proposed scan signature cannot protect our method as discussed in Section 6.

This paper is organized as follows: Section 2 introduces RSA encryption and decryption algorithms; Section 3 shows an algorithm of retrieving a secret key in an RSA circuit using intermediate values and explains problems to retrieve a secret key using a scan path; Section 4 proposes our scan-based side-channel attack method based on a *scan signature*; Section 5 demonstrates experimental results and performance analysis; Section 7 gives several concluding remarks.

2. RSA algorithm

RSA cryptography (Rivest et al., 1978) was made public in 1978 by Ronald Linn Rivest, Adi Shamir, Leonard Max Adleman. The RSA is known as the first algorithm which makes public-key cryptography practicable. It is commonly used to achieve not only encryption/decryption but also a digital signature and a digital authentication, so that most cryptography LSIs in the market implement and calculate the RSA cryptography.

The security of an RSA cryptography depends on the difficulty of factoring large numbers. To decrypt a ciphertext of an RSA cryptography will be almost impossible on the assumption that no efficient algorithm exists for solving it.

2.1 Encryption and decryption

An RSA algorithm encrypts a plaintext with a public key (n, e) and decrypts a ciphertext with a secret key (n, d). Let us select two distinct prime numbers p and q. We calculate n by

Algorithm 1 Binary method (MSB to LSB).

Input: c, d, and n.
Output: $c^d \bmod n$.
  $i = L - 1$.
  $m = 1$.
  while $i \geq 0$ **do**
    $m = m^2 \mod n$.
    if $d_i = 1$ **then**
      $m = m \times c \mod n$.
    end if
    $i = i - 1$.
  end while
  return m.

multiplying p by q, which is used as the modulus for both a public key and a secret key. To determine exponents of them, we calculate $\varphi(pq)$[1] for multiplying $(p-1)$ by $(q-1)$.

Let us select an integer e satisfying the conditions that $1 < e < \varphi(pq)$ and, e and $\varphi(pq)$ is coprime, where e is an exponent of a public key. Let us determine an integer d satisfying the congruence relation $de \equiv 1 \mod \varphi(pq)$. That is to say, the public key consists of the modulus n and the exponent e. The private key consists of the modulus n and the exponent d.

Let us consider that Alice secretly sends a message m to Bob. First, Alice receives his public key (n, e). Second, she calculates the ciphertext c with Equation 1.

$$c = m^e \mod n \tag{1}$$

Then Alice transmits c to Bob. Bob decrypts c by using his private key and receive her message m. Equation 2 represents a decryption computation.

$$m \equiv c^d \mod n \tag{2}$$

2.2 Binary method

The bit length of an RSA key must be more than 1,024 bits because its security depends on its key length. It is currently recommended that n be at least 2,048 bits long (Silverman, 2002). This means that the exponent d in Equation 2 is at least 1,024 bits long. When we decrypt a cyphertext, its computation amount becomes quite large without modification. Since modulo exponentiation dominates the execution time of decrypting a cyphertext, efficient algorithms have been proposed. The binary method (Stein, 1967), as shown in Algorithm 1, is one of the most typical exponent algorithms. In Algorithm 1, the exponent d is represented by $d = d_{L-1}2^{L-1} + d_{L-2}2^{L-2} + \cdots + d_1 2 + d_0$, where L shows the maximum key bit length. Fig. 1 shows an example of the binary method in case of $d = 1011_2$.

3. Scan-based attack against RSA

A scan path connects registers in an circuit serially and makes us access to them directly so that a tester can observe register values inside the circuit easily. A scan path model is shown

[1] $\varphi()$ is Euler's totient function.

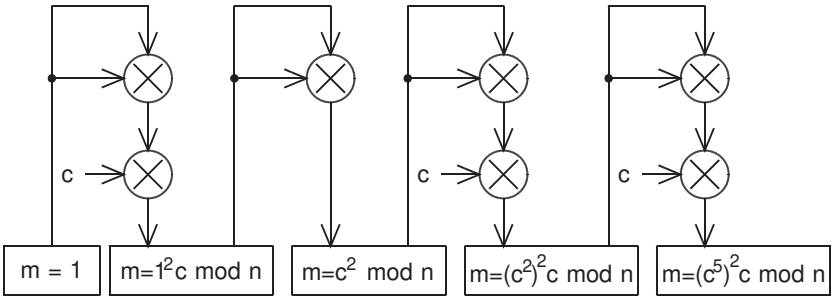


Fig. 1. Binary method example ($d = 1011_2$).

in Fig. 2. A scan path test is widely used in recent circuit implementations due to its testability and easiness of implementation.

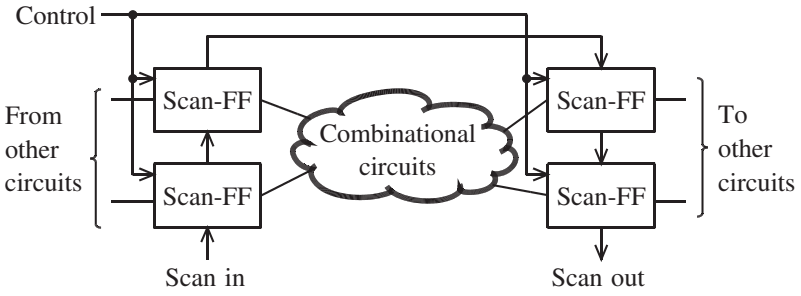


Fig. 2. Scan path model.

The purpose of a scan-based attack against RSA is to retrieve a secret exponent d from scanned data in an RSA circuit. Scan-based attack here requires several assumptions as in the previous researches in (Nara et al., 2009; 2011; Yang et al., 2004; 2006), which are summarized as shown below:

1. Attackers can encrypt/decrypt arbitrary data using the secret key on a target RSA circuit.
2. Attackers can obtain scanned data from a target RSA circuit.
3. Scanned data is not modified with compactors aimed at test efficiency.
4. Attackers know that the binary method in Algorithm 1 is used in a target RSA circuit.
5. Attackers also know the modulus n used in a target RSA circuit.[2]

In addition to these, they need to be able to predict the intermediate values of the binary method using an off-line simulation.

In this section, we explain the scan-based attack against an RSA circuit (Section 3.1) and its problems in a practical case (Section 3.2).

[2] Note that, since the public key consists of the modulus n and the public exponent e, attackers can easily know the modulus n.

3.1 Retrieving a secret exponent using intermediate values (Messerges et al., 1999)

In order to retrieve a secret exponent d, we have to solve the integer factorization in RSA. If the bit length of a secret exponent d is more than 1,024 bits or more than 2,048 bits, it is impossible to solve this problem within a realistic time. However, if we know all the "intermediate values" during the binary method shown in Algorithm 1, we can retrieve a secret exponent d in a polynomial time (Messerges et al., 1999).

Let $d = d_{L-1}2^{L-1} + d_{L-2}2^{L-2} + \cdots + d_1 2 + d_0$, where L is the maximum key bit length of d. Assume that all the intermediate values in Algorithm 1 are obtained. Let $m(i)$ be the intermediate value of m at the end of loop i in Algorithm 1. Assume also that $d_{L-1}, d_{L-2}, \cdots, d_{i+1}$ are already retrieved. An attacker tries to reveal the next bit d_i. In this case, $m(i)$ is equal to Equation 3 below, if and only if $d_i = 0$:

$$c^{\sum_{j=i+1}^{L-1} d_j 2^{j-i}} \mod n. \tag{3}$$

Similarly, $m(i)$ is equal to Equation 4 below, if and only if $d_i = 1$:

$$c^{\sum_{j=i+1}^{L-1} d_j 2^{j-i}+1} \mod n. \tag{4}$$

Based on the above discussion, we employ $SF(i)$ defined by Equation 5 as a *selective function* for RSA:

$$SF(i) = c^{\sum_{j=i+1}^{\ell-1} d_j 2^{j-i}+1} \mod n. \tag{5}$$

ℓ represents a significant key length, or key length in left-align representation, i.e., the secret exponent can be represented by

$$\begin{aligned} d &= d_{L-1}2^{L-1} + \cdots + d_1 2 + d_0 \Big|_{d_{L-1}=0,\ldots,d_\ell=0} \\ &= d_{\ell-1}2^{\ell-1} + \cdots + d_1 2 + d_0. \end{aligned} \tag{6}$$

When using the selective function for RSA above, we have to know in advance $d_{\ell-1}, d_{\ell-2}, \cdots, d_{i+1}$.

$SF(i) \neq SF(j)$ always holds true for $i \neq j$ for $0 \leq i, j \leq \ell - 1$. Given a message c and bit values of secret component $d_{\ell-1}, d_{\ell-2}, \cdots, d_{i+1}$, we assume that $d_i = 1$ and check whether $SF(i)$ appears somewhere in intermediate values. If it appears in them, we really determine d_i as one. If not, we determine d_i as zero.

Example 1. *Let us consider that the public key* $(n, e) = (101111001, 1011)$ *and the secret key* $(n, d) = (101111001, 10111)$. *The maximum key length L is 8 bits and the secret exponent* $d = 10111$, *i.e,* $d_7 = 0$, $d_6 = 0$, $d_5 = 0$, $d_4 = 1$, $d_3 = 0$, $d_2 = 1$, $d_1 = 1$, $d_0 = 1$. *We assume that we do not know d and a significant key length* ℓ. *The intermediate values in Algorithm 1 are summarized in Table 2 when we use a message* $c = 10011100$, *whose parameters are shown in Table 1.*

Now we try to retrieve the 8-bit secret exponent d using intermediate values.

First we try to retrieve the first bit $d_{\ell-1}$ $(i = \ell - 1)$. *We find* $d_{\ell-1} = 1$ *by the definition of a significant key length* ℓ. *Then* $SF(\ell - 1)$ *is calculated as* $SF(\ell - 1) = c = 10011100$. *Since* 10011100 *appears in Table 2, we confirm that* $d_{\ell-1}$ *is retrieved as one. Now we assume that the secret exponent* $d = \underline{\mathbf{1}}$. *We*

compare $m(\ell-1) = (c^1 \mod n) = 10011100$ *with the binary method result* 10001111. *Since they are not equal,* $d \neq 1$.

Next, we try to retrieve the second bit $d_{\ell-2}$ *(*$i = \ell - 2$*). We have already known that* $d_{\ell-1} = 1$. *We assume here that* $d_{\ell-2} = 1$. *In this case,* $SF(\ell-2)$ *is calculated as* $SF(\ell-2) = 11010$. *Since* 11010 *does not appear in Table 2, then* $d_{\ell-2}$ *is retrieved not as one but as zero, i.e.,* $d_{\ell-2} = 0$. *Now we assume that* $d = 1\underline{\mathbf{0}}$. *We compare* $m(\ell-2) = (c^{10} \mod n) = (m(\ell-1)^2 \mod n) = 11010000$ *with the binary method result* 10001111. *Since they are not equal,* $d \neq 10$.

Next, we try to retrieve the third bit $d_{\ell-3}$ *(*$i = \ell - 3$*). We have already known that* $d_{\ell-1} = 1$ *and* $d_{\ell-2} = 0$. *We assume here that* $d_{\ell-3} = 1$. *In this case,* $SF(\ell-3)$ *is calculated as* $SF(\ell-3) = 10000010$. *Since* 10000010 *appears in Table 2, then* $d_{\ell-3}$ *is retrieved as one, i.e.,* $d_{\ell-3} = 1$. *Now we assume that* $d = 10\underline{\mathbf{1}}$. *We compare* $m(\ell-3) = (c^{101} \mod n) = SF(\ell-3) = 10000010$ *with the binary method result* 10001111. *Since they are not equal,* $d \neq 101$.

Next, we try to retrieve the fourth bit $d_{\ell-4}$ *(*$i = \ell - 4$*). We have already known that* $d_{\ell-1} = 1$, $d_{\ell-2} = 0$ *and* $d_{\ell-3} = 1$. *We assume here that* $d_{\ell-4} = 1$. *In this case,* $SF(\ell-4)$ *is calculated as* $SF(\ell-4) = 100111$. *Since* 100111 *appears in Table 2, then* $d_{\ell-1}$ *is retrieved as one, i.e.,* $d_{\ell-4} = 1$. *Now we assume that* $d = 101\underline{\mathbf{1}}$. *We compare* $m(\ell-4) = (c^{1011} \mod n) = SF(\ell-4) = 100111$ *with the binary method result* 10001111. *Since they are not equal,* $d \neq 1011$.

We have already known that $d_{\ell-1} = 1$, $d_{\ell-2} = 0$, $d_{\ell-3} = 1$ *and* $d_{\ell-4=1}$. *We assume here that* $d_{\ell-5} = 1$. $SF(\ell-5)$ *is calculated as* $SF(\ell-5) = 10001111$ *(*$i = \ell - 5$*). Since* 10001111 *appears in Table 2, then* $d_{\ell-5}$ *is retrieved as one, i.e.,* $d_{\ell-5} = 1$. *Now we assume that* $d = 1011\underline{\mathbf{1}}$. *We compare* $m(\ell-5) = (c^{10111} \mod n) = SF(\ell-5) = 10001111$ *with the binary method result* 10001111. *Since they are equal to each other, we find that the secret exponent* d *is* 10111 *and a significant bit* ℓ *is five.*

3.2 Problems to retrieve a secret key using scan path

If we retrieve an L-bit secret exponent d using an exhaustive search, we have to try 2^L possible values to do it. On the other hand, the method explained in Section 3.1 retrieves a secret exponent one-bit by one-bit from MSB to LSB. It tries at most $2L$ possible values to retrieve an L-bit secret exponent. Further, the method just checks whether $SF(i)$ exists in the intermediate value $m(i)$ in Algorithm 1.

In order to apply this method to a scan-based attack, we have to know which registers store intermediate values, i.e., we have to know correspondence between scanned data and $SF(i)$.

However, scan paths are usually designed automatically by EDA tools so that nearby registers are connected together to shorten the scan path length. Only designers can know the correspondence between scanned data and registers and thus retrieved scanned data can be considered to be "random" for attackers. Therefore, it is very difficult to find out the values of $SF(i)$ in scanned data for attackers.

Messerges (Messerges et al., 1999) only shows the correspondence between intermediate values and a bit of a secret exponent. It does not indicate the method how to discover the intermediate value from scanned data. For that reason, its analysis method cannot directly apply to scan-based attacks against an RSA LSI.

We have to find out only $SF(i)$ somehow in the scanned data to retrieve a secret exponent d using the method in Section 3.1.

Maximum key length L	8 bits
Modulus n	101111001
Public exponent e	1011
Secret exponent d	10111

Table 1. Example parameters in Algorithm 1.

i	d_i	m^2	m
7	0	1	1
6	0	1	1
5	0	1	1
4	1	1	10011100
3	0	11010000	11010000
2	1	100011110	10000010
1	1	100111000	100111
0	1	1101	10001111

Table 2. Intermediate values at the end of i-th loop of Algorithm 1 (message $c = 10011100_2$).

4. Analysis scanned data

In order to solve the problem that attackers do not know the correspondence between registers of the scanned data and ones storing intermediate values during the binary method, we focus on the general property on scan paths: *a bit position of a particular register r in a scanned data when giving one input data is exactly the same as that when giving another input data.* This is clearly true, since a scan path is fixed in an LSI chip and the order of connected registers in its scan path is unchanged.

If we execute the binary method for each of N messages on an RSA circuit, a bit pattern of a *particular* bit position in scanned data for these N messages gives N-bit data. Based on the above property, this N-bit data may give a bit pattern of a particular bit in an intermediate value when we give each of these N messages to the RSA circuit.

We can calculate $SF(i)$ from the same N messages and $d_{\ell-1}$ down to d_0 of the secret exponent d by using an off-line simulation. By picking up a particular bit (LSB, for example) in each of $SF(i)$ values for N messages, we also have an N-bit data (see Fig. 3). If N is large enough, this N-bit data gives information completely unique to $SF(i)$. We can use this N-bit data as a *scan signature* SS_i to $SF(i)$ in scanned data.

Our main idea in this section is that we find out a scan signature SS_i to $SF(i)$ in scanned data (see Fig. 4) to retrieve the secret exponent d from $d_{\ell-1}$ down to d_0. If an N-bit scan signature SS_i appears in the scanned data for N messages, d_i is determined as one. If not, it is determined as zero.

In the rest of this section, we firstly propose a scan signature SS_i to $SF(i)$. Secondly we propose an overall method to retrieve a secret exponent d using scan signatures. Thirdly we analyze the probabilities of successfully retrieving a secret exponent by using our method.

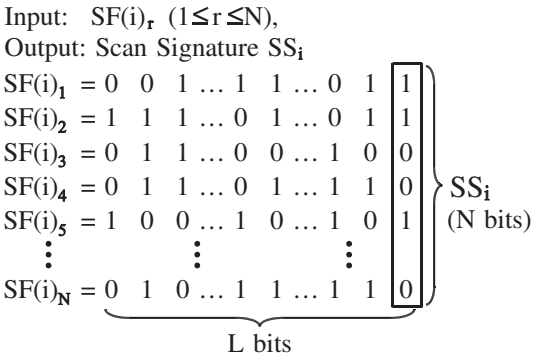


Fig. 3. Scan signature SS_i.

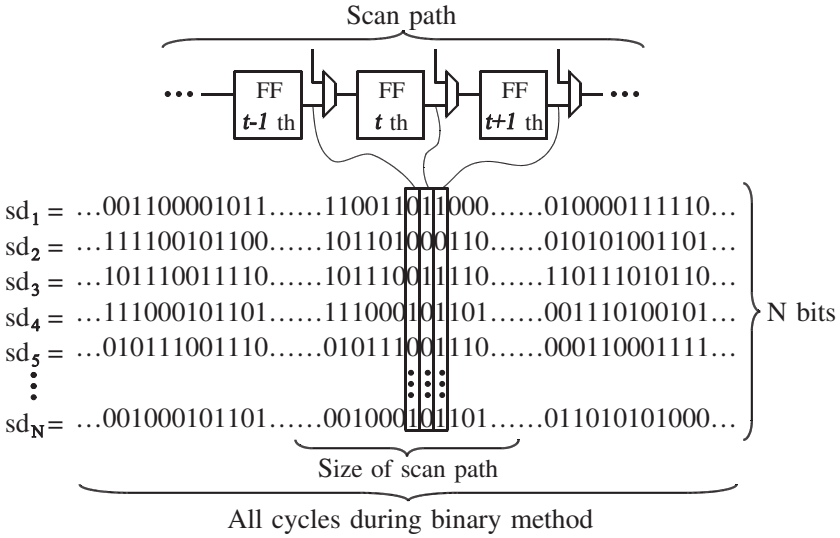


Fig. 4. Scanned data.

4.1 Calculating a scan signature to $SF(i)$

Assume that N messages $c_1, \cdots, c_N$ are given. Also assume that we have already known $d_{\ell-1}, \cdots, d_{i+1}$ for a secret exponent d. Let $SF(i)_r$ be the selective function for RSA when giving the message c_r for $1 \leq r \leq N$. Assuming that $d_i = 1$, we can calculate $SF(i)_r$ for $1 \leq r \leq N$.

Let us focus on a particular bit of $SF(i)_r$. If N is large enough, a set of these bits for $SF(i)_r$ $(1 \leq r \leq N)$ gives information unique to $SF(i)_r$. By using it, we can check whether $SF(i)_r$ are calculated or not in the target. As Fig. 3 shows, we define a *scan signature* SS_i to be a set of $SF(i)_r$ LSBs for the sake of convenience.

If SS_i appears in scanned data, d_i is determined as one. If not, d_i is determined as zero. After d_i is correctly determined, we can continue to determine the next bit of the secret exponent d in the same way.

Our proposed method has an advantage compared to conventional scan-based attacks (Yang et al., 2004; 2006). Our method is effective in the case of partial scan architecture. As long as a scan path includes at least 1-bit of each intermediate value, we can check whether the scan signature exists or not in the scanned data.

4.2 Scanned data analysis method

First we prepare N messages $c_1, \cdots, c_N$ and give them to an RSA circuit. For each of these messages, we obtain all the scanned data from the scan out of the RSA circuit until it outputs the binary method result. As Fig. 4 shows, the size of scanned data for each of these messages is ("scan path length" $\times$ "number of binary method cycles.")

Now we check whether a scan signature SS_i to $SF(i)$ appears in the obtained scanned data under the assumption that we do not know a secret exponent d in the RSA circuit as follows:

Step 1: Prepare N messages $c_1, c_2, \cdots, c_N$, where $c_r \neq c_s$ for $1 \leq r, s \leq N$ and $r \neq s$.

Step 2: Input c_r $(1 \leq r \leq N)$ into the target RSA circuit and obtain scanned data every one cycle while the binary method works, until the RSA circuit outputs the result. Let sd_r denote the obtained scanned data for the message c_r $(1 \leq r \leq N)$.

Step 3: From the definition, we have $d_{\ell-1} = 1$. Compare $m(\ell-1) = (c_1 \mod n)$ with its binary method result. If they are equal, then we find that the secret exponent d is one and stop. If not, go to the next step.

Step 4: Calculate $SF(\ell-2)_r$ assuming $d_{\ell-2} = 1$ for each c_r $(1 \leq r \leq N)$ and obtain the scan signature $SS_{\ell-2}$.

Step 5: Check whether the scan signature $SS_{\ell-2}$ exists in the scanned data $sd_1, \cdots, sd_N$, which includes the scanned data in all the cycles while the binary method runs. If it exists, then we can find out that $d_{\ell-2}$ is equal to 1, and if it does not exist, then we can find out that $d_{\ell-2}$ is equal to 0.

Step 6: Calculate $m(\ell-2) = ((c_1)^{d_{\ell-1}\times 2 + d_{\ell-2}} \mod n)$ and compare it with its binary method result. If they are equal, then we find that the secret exponent d is retrieved and terminate the analysis flow.

Step 7: We determine $d_{\ell-3}, d_{\ell-4}, \cdots$ in the same way as Step 4–Step 6 until the analysis flow is terminated at Step 6.

We show the example below to explain how the method above works.

Example 2. *As in Example 1, let us consider that the public key* $(n, e) = (101111001, 11)$ *and the secret key* $(n, d) = (101111001, 10111)$. *The maximum key length L is 8 bits and the secret exponent* $d = 10111_{10} = 10111_2$, *i.e,* $d_7 = 0$, $d_6 = 0$, $d_5 = 0$, $d_4 = 1$, $d_3 = 0$, $d_2 = 1$, $d_1 = 1$, $d_0 = 1$. *We assume that we do not know d and a significant key length* ℓ. *The parameters are shown in Table 1. Assume that the cycle counts of binary method are 16 and the size of the scan path is 128 in the target RSA circuit.*

(**Step 1**) *First we prepare* 8 *messages* $c_1, c_2, \cdots, c_8$, *where* $c_r \neq c_s$ *for* $1 \leq r, s \leq 8$ *and* $r \neq s$. *The target RSA circuit executes the binary method as in Table 2.*

(**Step 2**) *We input* c_r $(1 \leq r \leq 8)$ *into the target RSA circuit and obtain scanned data every one cycle while the binary method works, until the RSA circuit outputs the result. Let* sd_r *denote the obtained scanned data for the messages* c_r $(1 \leq r \leq 8)$. *The total size of scanned data is* $16 \times 128 = 2,048$ *(see Fig. 5).*

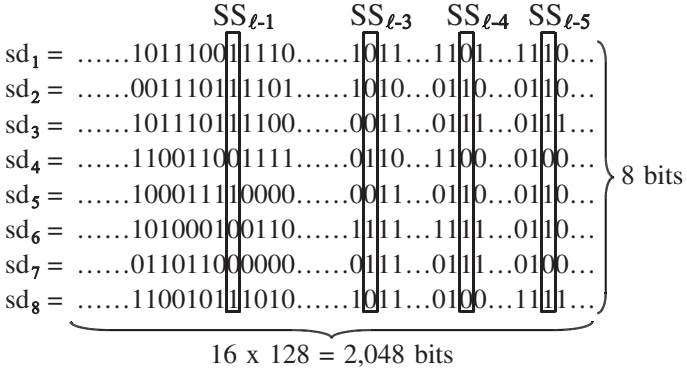


Fig. 5. Scanned data example.

(**Step 3**) *Let us start to determine* $d_{\ell-1}$. *We find* $d_{\ell-1} = 1$ *by the definition of* ℓ. *It is not necessary to check whether* $d_{\ell-1} = 1$ *or not, but we can check it as follows: we calculate* $SF(\ell-1)_r = c_r$ *for each* c_r $(1 \leq r \leq 8)$ *and obtain the scan signature* $SS_{\ell-1}$ *(see Fig. 6). As Fig. 6 (a) shows, the scan signature* $SS_{\ell-1}$ *becomes "11101001". Since we find out that the scan signature* $SS_{\ell-1}$ *exists in bit patterns of scanned data* sd_r $(1 \leq r \leq 8)$ *in Fig. 5, we confirm that* $d_{\ell-1}$ *is retrieved as one, i.e.,* $d_{\ell-1} = 1$. *Now we assume that* $d = \underline{\mathbf{1}}$. *We compare* $m(\ell-1) = ((c_1)^1 \mod n)$ *with its binary method result. In case they are not equal,* $d \neq 1$.

(**Step 4**, **Step 5**, **Step 6**, *and* **Step 7**) *Next let us determine* $d_{\ell-2}$. *We calculate* $SF(\ell-2)_r$ *assuming* $d_{\ell-2} = 1$ *for each* c_r $(1 \leq r \leq 8)$ *and obtain the scan signature* $SS_{\ell-2}$ *(see Fig. 6 (b)). As Fig. 6 (b) shows, the scan signature* $SS_{\ell-2}$ *becomes "01111100". Since we find out that the scan signature* $SS_{\ell-2}$ *does not exist in bit patterns of scanned data* sd_r $(1 \leq r \leq 8)$ *in Fig. 5, we can determine that* $d_{\ell-2}$ *is equal to zero, i.e.,* $d_{\ell-2} = 0$. *Now we assume that* $d = 1\underline{\mathbf{0}}$. *We compare* $m(\ell-2) = (m(\ell-1)^2 \mod n)$ *with its binary method result. In case they are not equal,* $d \neq 10$.

(**Step 4**, **Step 5**, **Step 6**, *and* **Step 7**) *Next let us determine* $d_{\ell-3}$. *We calculate* $SF(\ell-3)_r$ *assuming* $d_{\ell-3} = 1$ *for each* c_r $(1 \leq r \leq 8)$ *and obtain the scan signature* $SS_{\ell-3}$ *(see Fig. 6 (c)). As Fig. 6 (c) shows, the scan signature* $SS_{\ell-3}$ *becomes "00010110". Since we find out that the scan signature* $SS_{\ell-3}$ *exists in bit patterns of scanned data* sd_r $(1 \leq r \leq 8)$ *in Fig. 5, we can determine that* $d_{\ell-3}$ *is equal to one, i.e.,* $d_{\ell-3} = 1$. *Now we assume that* $d = 10\underline{\mathbf{1}}$. *We compare* $m(\ell-3) = (m(\ell-2)^2 \times c_1 \mod n) = SF(\ell-3)_1$ *with its binary method result. In case they are not equal,* $d \neq 101$.

(**Step 4**, **Step 5**, **Step 6**, *and* **Step 7**) *Next let us determine* $d_{\ell-4}$. *We calculate* $SF(\ell-4)_r$ *assuming* $d_{\ell-4} = 1$ *for each* c_r $(1 \leq r \leq 8)$ *and obtain the scan signature* $SS_{\ell-4}$ *(see Fig. 6 (d)). As Fig. 6 (d) shows, the scan signature* $SS_{\ell-4}$ *becomes "01101110". Since we find out that the scan signature* $SS_{\ell-4}$ *exists in bit patterns of scanned data* sd_r $(1 \leq r \leq 8)$, *we can determine that* $d_{\ell-4}$ *is equal to one, i.e.,* $d_{\ell-4} = 1$. *Now we assume that* $d = 101\underline{\mathbf{1}}$. *We compare* $m(\ell-4) = (m(\ell-3)^2 \times c_1 \mod n) = SF(\ell-4)_1$ *with its binary method result. In case they are not equal,* $d \neq 1011$.

(**Step 4**, **Step 5**, **Step 6**, *and* **Step 7**) *Finally let us determine* $d_{\ell-5}$. *We calculate* $SF(\ell-5)_r$ *assuming* $d_{\ell-5} = 1$ *for each* c_r $(1 \leq r \leq 8)$ *and obtain the scan signature* $SS_{\ell-5}$ *(see Fig. 6 (e)). As Fig. 6*

Input: $SF(\ell\text{-}1)_r$ $(1 \le r \le 8)$
Output: Scan signature $SS_{\ell\text{-}1}$

$SF(\ell\text{-}1)_1 = 1\ 0\ 1\ 1\ 0\ 0\ 1\ \boxed{1}$
$SF(\ell\text{-}1)_2 = 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1$
$SF(\ell\text{-}1)_3 = 0\ 0\ 1\ 0\ 1\ 1\ 0\ 1$
$SF(\ell\text{-}1)_4 = 0\ 1\ 0\ 1\ 0\ 0\ 1\ 0$
$SF(\ell\text{-}1)_5 = 1\ 0\ 1\ 0\ 1\ 1\ 0\ 1$
$SF(\ell\text{-}1)_6 = 0\ 0\ 0\ 1\ 0\ 0\ 1\ 0$
$SF(\ell\text{-}1)_7 = 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0$
$SF(\ell\text{-}1)_8 = 0\ 0\ 0\ 1\ 1\ 0\ 0\ 1$

$\} SS_{\ell\text{-}1}$

(a) Scan signature $SS_{\ell\text{-}1}$.

Input: $SF(\ell\text{-}2)_r$ $(1 \le r \le 8)$
Output: Scan signature $SS_{\ell\text{-}2}$

$SF(\ell\text{-}2)_1 = 0\ 0\ 0\ 0\ 1\ 0\ 1\ \boxed{0}$
$SF(\ell\text{-}2)_2 = 1\ 1\ 1\ 1\ 0\ 1\ 0\ 1$
$SF(\ell\text{-}2)_3 = 0\ 1\ 1\ 0\ 0\ 1\ 1\ 1$
$SF(\ell\text{-}2)_4 = 1\ 1\ 0\ 0\ 1\ 1\ 0\ 1$
$SF(\ell\text{-}2)_5 = 1\ 1\ 0\ 0\ 1\ 0\ 1\ 1$
$SF(\ell\text{-}2)_6 = 0\ 0\ 1\ 1\ 1\ 0\ 1\ 1$
$SF(\ell\text{-}2)_7 = 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0$
$SF(\ell\text{-}2)_8 = 0\ 1\ 1\ 1\ 0\ 1\ 0\ 0$

$\} SS_{\ell\text{-}2}$

(b) Scan signature $SS_{\ell\text{-}2}$.

Input: $SF(\ell\text{-}3)_r$ $(1 \le r \le 8)$
Output: Scan signature $SS_{\ell\text{-}3}$

$SF(\ell\text{-}3)_1 = 0\ 1\ 1\ 0\ 0\ 1\ 1\ \boxed{0}$
$SF(\ell\text{-}3)_2 = 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0$
$SF(\ell\text{-}3)_3 = 0\ 1\ 1\ 0\ 0\ 1\ 0\ 0$
$SF(\ell\text{-}3)_4 = 1\ 0\ 0\ 1\ 0\ 0\ 1\ 1$
$SF(\ell\text{-}3)_5 = 0\ 0\ 0\ 0\ 1\ 1\ 0\ 0$
$SF(\ell\text{-}3)_6 = 0\ 0\ 0\ 1\ 0\ 0\ 1\ 1$
$SF(\ell\text{-}3)_7 = 0\ 1\ 1\ 1\ 0\ 0\ 0\ 1$
$SF(\ell\text{-}3)_8 = 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0$

$\} SS_{\ell\text{-}3}$

(c) Scan signature $SS_{\ell\text{-}3}$.

Input: $SF(\ell\text{-}4)_r$ $(1 \le r \le 8)$
Output: Scan signature $SS_{\ell\text{-}4}$

$SF(\ell\text{-}4)_1 = 1\ 1\ 0\ 0\ 1\ 0\ 1\ \boxed{0}$
$SF(\ell\text{-}4)_2 = 0\ 1\ 0\ 1\ 1\ 1\ 0\ 1$
$SF(\ell\text{-}4)_3 = 1\ 1\ 1\ 0\ 0\ 1\ 1\ 1$
$SF(\ell\text{-}4)_4 = 0\ 1\ 1\ 0\ 0\ 1\ 0\ 0$
$SF(\ell\text{-}4)_5 = 1\ 1\ 0\ 1\ 1\ 0\ 1\ 1$
$SF(\ell\text{-}4)_6 = 1\ 0\ 1\ 1\ 0\ 0\ 1\ 1$
$SF(\ell\text{-}4)_7 = 0\ 1\ 1\ 0\ 0\ 1\ 1\ 1$
$SF(\ell\text{-}4)_8 = 1\ 0\ 1\ 1\ 0\ 0\ 0\ 0$

$\} SS_{\ell\text{-}4}$

(d) Scan signature $SS_{\ell\text{-}4}$.

Input: $SF(\ell\text{-}5)_r$ $(1 \le r \le 8)$
Output: Scan signature $SS_{\ell\text{-}5}$

$SF(\ell\text{-}5)_1 = 0\ 0\ 1\ 0\ 1\ 1\ 1\ \boxed{1}$
$SF(\ell\text{-}5)_2 = 0\ 0\ 0\ 1\ 1\ 0\ 0\ 1$
$SF(\ell\text{-}5)_3 = 0\ 0\ 1\ 0\ 0\ 1\ 0\ 1$
$SF(\ell\text{-}5)_4 = 0\ 1\ 1\ 1\ 0\ 1\ 0\ 0$
$SF(\ell\text{-}5)_5 = 0\ 1\ 1\ 1\ 1\ 0\ 1\ 1$
$SF(\ell\text{-}5)_6 = 1\ 0\ 1\ 0\ 0\ 0\ 0\ 1$
$SF(\ell\text{-}5)_7 = 0\ 1\ 1\ 0\ 1\ 1\ 1\ 0$
$SF(\ell\text{-}5)_8 = 1\ 0\ 1\ 1\ 1\ 0\ 1\ 1$

$\} SS_{\ell\text{-}5}$

(e) Scan signature $SS_{\ell\text{-}5}$.

Fig. 6. Example of scan signatures.

(e) shows, the scan signature $SS_{\ell-5}$ becomes "11101101". Since we find out that the scan signature $SS_{\ell-5}$ exists in bit patterns of scanned data sd_r $(1 \le r \le 8)$, we can determine that $d_{\ell-5}$ is equal to one, i.e., $d_{\ell-5} = 1$. Now we assume that $d = 1011\underline{\mathbf{1}}$. We compare $m(\ell-5) = (m(\ell-4)^2 \times c_1 \mod n) = SF(\ell-5)_1$ with its binary method result. In case they are equal to each other, we find that the secret exponent d is 10111 *and a significant bit ℓ is five.*

4.3 Possibility of successfully retrieving a secret key

Given that the scan size is α bits and the cycle counts to obtain the binary method result is T. Assume that scanned data are completely random data.

Even though $SF(i)_r$ for $1 \leq r \leq N$ is not calculated in the target RSA circuit, its scan signature may exist in scanned data. When $\alpha T < 2^N$, the probability that the scan signature SS_i to $SF(i)_r$ exists in somewhere in bit patterns of scanned data sd_r $(1 \leq r \leq N)$ is $\alpha T/2^N$ despite we do not calculate $SF(i)_r$.

Sufficiently large N can decrease the probability that we mistakenly find out the scan signature SS_i in scanned data. For instance, if α is 3,072, T is 1,024, and N is 30[3], then the probability that we mistakenly find out the scan signature SS_i in scanned data is $3,072 \times 1,024/2^{30} \simeq 2.93 \times 10^{-3}$. If α is 6,144, T is 2,048, and N is 35, then the probability that we mistakenly find out the scan signature SS_i in scanned data is $6,144 \times 2,048/2^{35} \simeq 3.66 \times 10^{-4}$.

5. Experiments and analysis

We have implemented our analysis method proposed in Section 4 in the C language on Red Hat Enterprise Linux 5.5, AMD Opteron 2360SE 2.5GHz, and 16GB memories and performed the following experiments:

1. First, we have generated secret exponents randomly. Thousand of them have a bit length of 1,024 and 2,048, respectively. The other hundred of them have a bit length of 4,096.
2. Next, we have given each of the secret exponents into the target RSA circuit based on Algorithm 1 and obtained scanned data. The target RSA circuit obtains binary method results in 1,024 cycles for a 1,024-bit secret exponent, in 2,048 cycles for a 2,048-bit secret exponent, and in 4,096 cycles for a 4,096-bit secret exponent. Scan path length for a 1,024-bit secret exponent is 3,072 bits, that for a 2,048-bit secret exponent is 6,144 bits, and that for a 4,096-bit secret exponent is 12,192 bits. Then total size of the obtained scanned data for 1,024-bit secret exponent is $3,072 \times 1,024 = 3,145,728$ bits, that for 2,048-bit secret exponent is $6,144 \times 2,048 = 12,582,912$ bits, and that for 4,096-bit secret exponent is $12,192 \times 4,096 = 49,938,432$ bits
3. Finally, we have retrieved each of the secret exponents by our proposed analysis method using the obtained scanned data.

Fig. 7 and Table 4 show the results. Fig. 7 shows the number N of required messages to retrieve each secret exponent when giving each of the secret exponents. For example, the 4th 1,024-bit secret exponent is shown in Table 3. In order to retrieve this secret exponent, we need 29 messages, i.e., $n = 29$. In this case, we can successfully retrieve the 4th secret exponent using 29 messages but fail to retrieve it using 28 messages or less.

Throughout this experiment, the required number of messages is approximately 29.5 on average for 1,024-bit secret exponents and is approximately 32 for 2,048-bit secret exponents and is approximately 37 for 4,096-bit secret exponents. A running time is 98.3 seconds to retrieve a 1,024-bit secret exponent and 634.0 seconds to retrieve a 2,048-bit secret exponent.

[3] These values are derived from the experiments in Section 5.

	4-th secret exponent d
1,024 bits	0x3AD29CF2FC6CB6B0C010B17DF98C5081 4E4585225AC42E8ECB7BB1847498D62F BA696CDD226EE9195F4E58A89321721F 021C4511E6C994301363706058FF3765 E29EEBA03E370A201BA5B60A356682A5 1D05EE10DF8CB75D7B4578B3D29A515E 2F86DEC487AB6BCD88C7351908D71851 6C11B2419BD8C05739214E6CF44D12F

Table 3. Secret exponent example.

Key bit length *bit*	1,024	2,048	4,096
# of retrieving secret exponents	1,000	1,000	100
# of required messages (average)	29.5	32	37

Table 4. Experimental results.

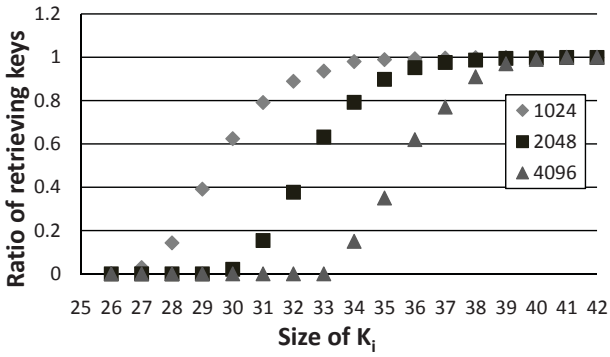


Fig. 7. Number of required messages to retrieve secret exponents.

6. Discussions

We consider secure scan architecture proposed so far against our proposed scan-based attack.

Firstly, the secure scan architecture proposed in (Sengar et al., 2007) cannot protect our proposed method from retrieving a secret key. (Sengar et al., 2007) inserts some inverters into a scan path to invert scanned data. However, since inverted positions of scanned data are always fixed, the value of a 1-bit register sequence is only changed to its inverted value. By checking whether SS_i or inverted SS_i exist in the scanned data, our proposed method can easily make it ineffective.

Inoue's secure scan architecture (Inoue et al., 2009) adds unrelated data to scanned data to confuse attackers. A sequence of scanned data to which unrelated data are added is fixed and it is not always true that they confuse all the bits to protect the scanned data in order to reduce area overhead. If the register storing scan signature SS_i is not confused, our proposed method can easily make it ineffective, too.

Secondly, (Chandran & Zhao, 2009; Gomułkiewicz et al., 2006; Hely et al., 2005; 2006; 2007; Lee et al., 2006; 2007; Paul et al., 2007; Yang et al., 2006) require authentication to transfer

between system mode and test mode, and their security depends on authentication methods. If authentication would be broken-through and attackers could obtain scanned data, a secret key in an RSA circuit could be retrieved by using our proposed method. We consider that authentication strength is a different issue from the purpose of this chapter.

Finally, (Mukhopadhyay, et al.; Sengar et al., 2007; Shi el al., 2008) use a compactor so as not to output scanned data corresponding to registers directly. (Doulcier el al., 2007) proposes AES-based BIST, whereby there is no need for scan path test. However, applying these methods effectively to an RSA circuit is quite unclear because these methods are implemented only on an AES circuit or just on a sample circuit not for cryptography.

7. Concluding remarks

Our proposed scan-based attack can effectively retrieve a secret key in an RSA circuit, since we just focus on the variation of 1-bit of intermediate values named a scan signature. By monitoring it in the scan path, we can find out the register position specific to intermediate values. The experimental results demonstrate that a 1,024-bit secret key can be retrieved by using 29.5 messages, a 2,048-bit secret key by using 32 input, and a 4,096-bit secret key can be retrieved by using 37 messages.

In the future, we will develop a new scan-based side-channel attack against compressed scan data for RSA. In this paper, we only pick up one RSA LSI implementation but there can be other implementations available such as in (Miyamoto et al., 2008). We will attack these RSA implementations and successfully retrieve a secret key. Developing countermeasures against the proposed scan-based side-channel attacking method is another future work.

8. References

Boneh, D.; DeMillo, R. A. & Lipton, R. J. (1997). On the importance of checking cryptographic protocols for faults, *Proceedings of Advances in Cryptology - EUROCRYPTO '97*, Lecture Notes in Computer Science, Vol. 1233, pp. 37–51.

Brier, E.; Clavier, C. & Olivier, F. (2004). Correlation power analysis with a leakage model, *Proceedings of Cryptography Hardware Embedded Systems 2004*, Lecture Notes in Computer Science, Vol. 3156, pp. 16–29.

Chandran, U. & Zhao, D. (2009). SS-KTC: a high-testability low-overhead scan architecture with multi-level security integration, *Proceedings of 27th IEEE VLSI Test Symposium*, pp. 321–326.

Doulcier, M.; Flottes, M. L. & Rouzeyre, B. (2007). AES-based BIST: self-test, test pattern generation and signature analysis, *Proceedings of 25th IEEE VLSI Test Symposium*, pp. 94–99.

Gomułkiewicz, M.; Nikodem, M. & Tomczak, T. (2006). Low-cost and universal secure scan: a design-architecture for crypto chips, *Proceedings of International Conference on Dependability of Computer Systems*, pp. 282–288.

Hely, D.; Bancel, F.; Flottes, M. L. & Rouzeyre, B. (2005), Test control for secure scan designs, *Proceedings of European Test Symposium*, pp. 190–195.

Hely, D.; Bancel, F.; Flottes, M. L. & Rouzeyre, B. (2006). Secure scan techniques: a comparison, *Proceedings of 12th IEEE International On-Line Testing Symposium*, pp. 119–124.

Hely, D.; Bancel, F.; Flottes, M. L. & Rouzeyre, B. (2007). Securing scan control in crypto chips, *Journal of Electron Test*, pp. 457–464.

Inoue, M.; Yoneda, T.; Hasegawa, M. & Fujiwara, H. (2009). Partial scan approach for secret information protection, *Proceedings of European Test Symposium*, pp. 143–148.

Kocher, P. C. (1996). Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems, *Proceedings of Advances in Cryptology - Crypto '96*, Lecture Notes in Computer Science, Vol. 1109, pp. 104–113.

Koblitz, N. (1987). Elliptic curve cryptosystems, *Mathematics of Computation*, Vol. 48, pp. 203–209.

Kocher, P. C.; Jaffe, J. & Jun, B. (1999). Differential power analysis, *Proceedings of 19th Annual International Cryptology Conference on Advances in Cryptology*, pp. 388–397.

Lee, J.; Tehranipoor, M. & Plusquellic, J. (2006). A low-cost solution for protecting IPs against scan-based side-channel attacks, *Proceedings of 24th IEEE VLSI Test Symposium*, pp. 94–99.

Lee, J.; Tehranipoor, M.; Patel, J. & Plusquellic, J. (2007). Securing designs against scan-based side-channel attacks, *IEEE Transactions on Dependable and Secure Computing*, pp. 325–336.

Messerges, T. S.; Dbbish, E. A. & Sloan, R. H. (1999). Power analysis attacks of modular exponentiation in smartcards, *Proceedings of Workshop on Cryptographic Hardware and Embedded Systems*, Lecture Notes in Computer Science, Vol. 1717, pp. 144–157.

Miller, V. (1986). Uses of elliptic curves in cryptography, *the Advances in Cryptology*, ed. H. Williams, pp. 417–426.

Miyamoto, A.; Homma, N.; Aoki, T. & Satoh, A. (2008). Systematic design of high-radix montgomery multipliers for RSA processors, *Proceedings of IEEE International Conference on Computer Design (ICCD 2008)*, pp. 416–421.

Mukhopadhyay, D.; Banerjee, S.; RoyChowdhury, D. & Bhattacharya, B. B. (2005). CryptoScan: a secured scan path architecture, *Proceedings of 14th Asian Test Symposium*, pp. 348–358.

Nara, R.; Togawa, N.; Yanagisawa, M. & Ohtsuki, T. (2009). A scan-based side-channel attack based on scan signatures for AES cryptosystems, *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E92–A, No. 12, pp. 3229–3237.

Nara, R.; Togawa, N.; Yanagisawa, M. & Ohtsuki, T. (2011). Scan vulnerability in elliptic curve cryptosystems, *IPSJ Transactions on System LSI Design Methodology*, Vol. 4, Sep., pp. 47–59.

Paul, S.; Chakraborty, R. S. & Bhunia, S. (2007). Vim-scan: a low overhead scan design approach for protection of secret key in scan-based secure chips, *Proceedings of the 25th IEEE VLSI Test Symposium* pp. 455–460.

Rivest, R. L.; Shamir, A. & Adelman, L. (1978). A method for obtaining digital signature and public-key cryptosystems, *Communications of the ACM*, Vol. 21, pp. 120–126.

Schramm, K.; Wollinger, T. & Paar, C. (2003). A new class of collision attacks and its application to DES, *T. Johansson (ed.) FSE 2003*, Lecture Notes in Computer Science, Vol. 2887, pp. 206–222.

Sengar, G.; Mukhopadhyay, D. & Chowdhury, D. R. (2007). Secured flipped scan-path model for crypto-architecture, *IEEE Transactions on Very Large Scale Integration System*, Vol. 26, No. 11, pp. 2080–2084.

Sengar, G.; Mukhopadhyay, D. & RoyChowdhury, D. (2007). An efficient approach to develop secure scan tree for crypto-hardware, *Proceedings of 15th International Conference of Advanced Computing and Communications*, pp. 21–26.

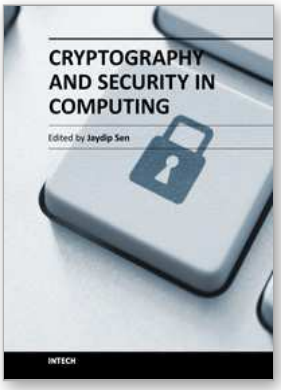
Shi, Y.; Togawa, N.; Yanagisawa, M. & Ohtsuki, T. (2008). A secure test technique for pipelined advanced encryption standard, *IEICE Transactions on Information and Systems*, Vol. E91–D, No. 3, pp. 776–780.

Silverman, R. D. (2002). Has the RSA algorithm been compromised as a result of Bernstein's Paper?
http://www.rsa.com/rsalabs/node.asp?id=2007, RSA Laboratories, April 8.

Stein, J. (1967). Computational problems associated with Racah algebra, *Journal of Computational Physics*, Vol. 1, pp. 397–405.

Yang, B.; Wu, K. & Karri, R. (2004). Scan based side-channel attack on dedicated hardware implementations of data encryption standard, *Proceedings of The International Test Conference*, pp. 339–344.

Yang, B.; Wu, K. & Karri, R. (2006). Secure scan: a design-for-test architecture for crypto chips, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 25, No. 10, pp. 2287–2293.



Cryptography and Security in Computing

Edited by Dr. Jaydip Sen

ISBN 978-953-51-0179-6
Hard cover, 242 pages
Publisher InTech
Published online 07, March, 2012
Published in print edition March, 2012

The purpose of this book is to present some of the critical security challenges in today's computing world and to discuss mechanisms for defending against those attacks by using classical and modern approaches of cryptography and other defence mechanisms. It contains eleven chapters which are divided into two parts. The chapters in Part 1 of the book mostly deal with theoretical and fundamental aspects of cryptography. The chapters in Part 2, on the other hand, discuss various applications of cryptographic protocols and techniques in designing computing and network security solutions. The book will be useful for researchers, engineers, graduate and doctoral students working in cryptography and security related areas. It will also be useful for faculty members of graduate schools and universities.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821