Security improvement for NTRU by using Error Analysis and Detection Procedures

Naglaa F. Saudy

Phy. And Applied Math. Dept. Faculty of Engineering, Helwan University, Cairo,Egypt

0020126260485, naglaa_eng2003@yahoo.com

Ahmed F. Shalash

Comm. Dept, Faculty of Engineering, Cairo University, Giza, Egypt.

shalash@ieee.org

Abdel-karim S.O. Hassan

Dept. of Eng. Math.& Physics, Faculty of Engineering, Cairo University, Giza, Egypt.

Asho_hassan@yahoo.com

ABSTRACT

In this paper, fault detection scheme is introduced to improve the security and reliability of the NTRU in realistic environments, such as operating over a network. As the NTRU will be working in real networks, which have their own set of transient errors, handling such errors in analyzing the NTRU becomes a must. A single transient error occurring during the NTRU encryption (or decryption) process will likely result in a large number of errors in the encrypted/decrypted data. Such faults must be detected before sending data to avoid the transmission and use of erroneous data. Concurrent fault detection is important not only to protect the encryption/decryption process from random faults, but it will also protect the decryption circuitry from an attacker who may maliciously inject faults in order to find the secret key. We first describe the effects that faults may have on NTRU security while operating over a network by analyzing the propagation of such faults to the outputs. We then present two fault detection schemes: The first is a redundancy-based scheme while the second uses an error detection code. The latter present fault detection schemes by using an error detecting code, using one parity bit. We will add the parity bits to the polynomials and disable the device output when any of these parity checks are violated. This fault detection scheme has more than 99% coverage fault detection in the encryption process and in the first part in the decryption process. In the second part in the decryption process, it has more than 67 % coverage fault detection. We can increase this ratio to high fault detection by adjusting the choices of the parameters of the NTRU network security system.

1. INTRODUCTION

Cryptography is the science of writing in secret code. Nth Degree Truncated Polynomial Ring Units, known as NTRU cryptography, was developed in 1996 by three mathematicians: Jeffrey Hoffstein, Joseph H. Silverman, and Jill Pipher. Later these three mathematicians in addition to Daniel Lieman founded the NTRU Cryptosystems, Inc, Boston, USA. [1] The main target was speeding up the encryption process by using NTRU. In 2009, NTRU Cryptosystem has been approved for standardization by the Institute of Electrical and Electronics Engineers (IEEE) [1]. NTRU is a lattice-based alternative to RSA and ECC and is based on the shortest vector search problem in a lattice. Thus making it a public key cryptosystem not based on factorization or discrete logarithmic problems. NTRU has many advantages over the other methods of encryption like RSA and DES [2]. For example, NTRU is a more efficient encryption and decryption, in both hardware and software implementations [2]. It has much faster key generation allowing the use of “disposable" keys (because keys are computationally “cheap" to create). It uses small memory size so it allows using it in applications such as mobile devices and Smart-cards. NTRU is also secure against chosen plaintext attacks as it is probabilistic cryptosystem [3]. NTRU is most naturally described using convolution polynomial rings, but the underlying hard mathematical problem can also be interpreted as SVP or CVP in a lattice [3]. Fault detection used to detect errors before sending data to avoid the transmission and use of erroneous data [4]. Moreover, fault detection is a desirable property for preventing malicious attacks [5], aimed at extracting sensitive information, like the secret key, from the device. Fault detection techniques are applied to several method of complex and non homogeneous cryptography like AES [6] [7]. In [7] a simple but efficient error detection code for AES is developed and evaluated which leads to very efficient and high coverage fault detection. In this paper, we present a study of the error propagation behavior of the encryption or decryption process. Then we discuss two fault detection schemes which increase NTRU security: The first is a redundancy-based scheme while the second uses an error detection code. The latter present fault detection schemes by using an error detecting code, using one parity bit. The paper is organized as follows: In Section 2, a brief overview of the NTRU algorithm is presented, including the implementation details which are necessary for understanding our proposed error detection schemes. Section 3 describes the analysis of error propagation in the encryption and decryption units for a simple single bit transient fault. This analysis allows us to obtain a rather comprehensive picture of the general behavior of NTRU in the presence of faults. Section 4 describes two fault detection algorithms. The first is a redundancy-based technique, while the second is based on exploiting error detecting codes, properly organized so as to fit NTRU. Finally, Section 5 concludes the paper. Appendices A, B and C outline several mathematical proofs for the error detection codes which are proposed in Section 3.

2. THE NTRU ALGORITHEM

In this section we describe the NTRU public key cryptosystem. We begin by fixing an integer $N \geq 1$ and two moduli p and q, and we let R, R_p, and R_q be the convolution polynomial rings

$$R = \frac{Z[x]}{(x^N - 1)} \qquad R_p = \frac{(Z/pZ)[x]}{(x^N - 1)} \qquad R_q = \frac{(Z/qZ)[x]}{(x^N - 1)}$$

We may view a polynomial $a(x) \in R$ as an element of R_p or R_q by reducing its coefficients modulo p or q. We make various assumptions on the parameters N, p and q. In particular, we require that N be prime and that $gcd(N, q) = gcd(p, q) = 1$ and $q > (6d + 1)p$. The parameters N, p and q are chosen in this way because if N or p is divisible by q then it is very easy for an attacker to decrypt the message without knowing the private key. And the parameter N is chosen to be prime because if it is not prime the attacker can recover the private key by solving a lattice problem in dimension N, rather than in dimension $2N$. The condition $q > (6d + 1)p$ ensures that decryption never fails.

For any positive integers d_1 and d_2, we let a(x) has

$$T(d_1, d_2) = a(x) \in R : \left\{ \begin{array}{l} \text{a(x) has } d_1 \text{ coefficients equal to 1,} \\ \text{a(x) has } d_2 \text{ coefficients equal to -1,} \\ \text{a(x) has all other coefficients equal to 0} \end{array} \right\}$$

Polynomials in $T(d_1, d_2)$ are called *ternary polynomials*. They are analogous to *binary polynomials*, which have only *0's* and *1's* as coefficients.

Let's assume that Alice (or some trusted authority) chooses public parameters (N, p, q, d). Alice's private key consists of two randomly chosen polynomials

$$f(x) \in T(d+1, d) \qquad \text{and} \qquad g(x) \in T(d, d) \tag{1}$$

Alice computes the inverses

$$f_q(x) = f(x)^{-1} \text{ in } R_q \quad \text{and} \qquad f_p(x) = f(x)^{-1} \text{ in } R_p. \tag{2}$$

Alice chooses $f(x)$ in $T(d + 1, d)$, rather than in $T(d, d)$, because elements in $T(d, d)$ never have inverses in R_q.

Alice next computes

$$h(x) = F_q(x) * g(x) \text{ in } Rq. \tag{3}$$

The polynomial *h(x)* is Alice's public key. Her private key, which she'll need to decrypt messages, is the pair *(f(x), F_p(x))*. Alternatively, Alice can just store *f (x)* and compute *F_p(x)* when she needs it.

Bob's plaintext is a polynomial *m(x)* $\in R$ whose coefficients are between $-\frac{1}{2p}$ and $\frac{1}{2p}$. Bob chooses a random polynomial (ephemeral key) *r(x)* $\in T(d, d)$ and computes

$$e(x) \equiv ph(x) * r(x) + m(x) (\bmod q). \tag{4}$$

Bob's cipher text *e(x)* is in the ring R_q.

On receiving Bob's cipher text, Alice starts the decryption process by computing

$$a(x) \equiv f(x) * e(x) (\bmod q) \tag{5}$$

She then center lifts a(*x*) to an element of *R* and does a mod *p* computation,

$$b(x) = Centerlift\ (a(x)) \tag{6}$$

$$m(x) \equiv F_p(x) * b(x) (\bmod p) \tag{7}$$

Table 1 shows some suggested choices for (*N; p; q*) for different security levels of the original NTRU encryption algorithm [8].

Table 1. The parameter sets for NTRU in [8]

	N	**P**	**Q**
Moderate Security	167	3	128
High Security	263	3	128
Highest Security	503	3	256

3. ERROR ANALYSIS:

In this section, the error propagation behavior of the encryption or decryption process is studied.

The purpose of this study is to understand the effect of a fault occurring during the execution of the algorithm on the final result. This is an important first step when developing fault detection schemes. For simplicity, we will study the effect of single faulty bits inserted in each step.

In the encryption process due to the nonlinearity as shown in Eq. (4) it is possible to see that a faulty bit inserted in the encryption causes a large number of erroneous bits in the final encrypted data.

In the decryption process we have two operations the first operation in Eq. (5) the injection of a single faulty bit at the input will cause only single faulty bit at the output but in the second operation in Eq. (6) fault spreads considerably.

4. FAULT DETECTION TECHNIQUES

The propose of this paper is to detect a fault in order to prevent the transmission and use of incorrect data. This issue is important as any hardware implementation of NTRU is bound to be complex and, consequently, likely to be subject to fault occurrences. In this section, two techniques for fault detection are presented. The first technique is based on redundancy. The second one is based on error detecting codes.

4.1 Redundancy-Based Technique

The redundancy-based solution for implementing fault detection in the encryption module is based on the idea of performing a test decryption immediately after the encryption and then checking whether the original data block is obtained. The overhead is close to 100 percent. The time penalty in either of these two cases is the time required to decrypt a data block, plus the time required for the comparison.

4.2 Error Detecting Codes

Error detecting codes (EDCs) have been widely used in practice. EDCs may at first seem unsuitable for implementing error detection in NTRU since NTRU is strongly nonlinear algorithm and because errors spread quickly over the data block (Sections 3). In this section, an efficient EDC scheme for NTRU will be described and evaluated. It achieves a high level of fault coverage at a limited hardware overhead cost.

To implement this coding scheme, it is necessary to develop, for each operation in the encryption and decryption process, a method for predicting the output parity, given the input operation and the input parity. We then need to schedule checkpoints during the encryption and decryption process.

We next describe the structure of the parity bits' prediction and checking scheme. Further details about the coding scheme, are included in Appendices A through C.

We present in what follows our proposed parity prediction algorithms for the individual operations.

4.2.1 Encryption process:

The prediction of the output parity bits of the encryption process is complex. The detailed solution is described in Appendix A.

To check the parity bits and generate a parity error flag, we need a set of parity generators and comparators which will compare the predicted parity bits to the generated parity bits

We will add parity bit for m(x) equal $(\sum_{i=1}^{n} m_i) \bmod q$ then after the step of calculating e(x) we will add the parity bit to $(p * (\sum_{i=1}^{n} r_i) \bmod q * (\sum_{i=1}^{n} h_i) \bmod q) \bmod q$ (predicted parity bits)and compare by $(\sum_{i=1}^{n} e_i) \bmod q$ (generated parity bits)if they are equal then there is no errors (except the case when the summation of errors is multiples of q.) The probability of detecting errors is greater than or equal $\frac{q-1}{q}$ where q is greater than or equal 128 (Table 1)so it has 99% coverage fault detection.

4.2.2 Decryption process:

In the decryption process we have two processes (see section 2). In the first process we will work with Eq. 5 by adding parity bit for a(x) equal $(\sum_{i=1}^{n} a_i) \bmod q$ (predicted parity bits) then after the step of calculating b(x) we will compare the parity bit by $(\sum_{i=1}^{n} b_i) \bmod q$ (generated parity bits) if they are equal then there is no errors (except the case when the summation of errors is multiples of q.) The probability of detecting errors is greater than or equal $\frac{q-1}{q}$ where q is greater than or equal 128 (Table 1) so it has 99% coverage fault detection. The detailed solution is described in Appendix B.

In the second process we will add parity bit for b(x) equal $(\sum_{i=1}^{n} b_i) \bmod q$ then after the step of calculating m(x) we will multiply the parity bit by $(\sum_{i=1}^{n} f_i) \bmod p$ and compare by $(\sum_{i=1}^{n} m_i) \bmod p$ if they are equal then there is no errors (except the case when the summation of errors is multiples of p.) The probability of detecting errors is greater than or equal $\frac{p-1}{p}$ where p is greater than or equal 3 (Table 1) so it has 67% coverage fault detection. We can increase this ratio to high fault detection by increasing the choices of the parameter p. The detailed solution is described in Appendix C.

5. CONCLUSIONS

An analysis of the behavior of the NTRU cryptosystem in the presence of faults has been carried out. We presented the effect of a single error on the NTRU encryption (or decryption) process and we found that it will cause a large number of errors in the encrypted/decrypted data. Two proposals for fault detection have been presented in this paper. The second one, which is based on the use of parity codes, exhibits good fault coverage, limited hardware overhead cost. We presented fault detection schemes by using an error detecting code (using parity bit). We added parity bits to the polynomials and disable the device output when any of these parity checks are violated. The probability of detecting errors equal $\frac{q-1}{q}$ in the case the encryption process and in the first part of the decryption process. This probability is large because $q > (6d+1)p$. In the case of the second part of decryption process the probability of detecting errors equal $\frac{p-1}{p}$ and this is may be a small. We can increase it by increasing the value of p.

APPENDIX A

PARITY PREDICTION IN ENCRYPTION PROCESS

In this appendix, we describe the formal construction of the parity prediction scheme for the encryption process (see Section 2).

The parity operator $p()$ is a function of the type:

$$p : GF(r^n) \rightarrow GF(r),$$

calculated as

$$\sum_{i=0}^{n-1} a_i x^i \rightarrow \sum_{i=0}^{n-1} a_i \qquad \text{for any } n \geq 1$$

In other words, the coefficients a_i of the polynomial A(x) are added modulus r. Clearly, the parity is a linear operator, i.e. $p(A(x) \oplus B(x)) = p(A(x)) + p(B(x))$

The following provides the prove for predicting the parity bits for the outputs of the encryption process, assuming that the inputs and their corresponding parity bits are known.

Given the following polynomials with coefficients over $GF(q^n)$, for some $n \geq 1$,

$$m(x) = \sum_{i=1}^{n} m_i x^i \qquad r(x) = \sum_{i=1}^{n} r_i x^i \qquad h(x) = \sum_{i=1}^{n} h_i x^i$$

We will add parity bit for *m(x)* equal $(\sum_{i=1}^{n} m_i) \bmod q$ then after the step of calculating *e(x)* we will add the parity bit to $(p * (\sum_{i=1}^{n} r_i) \bmod q * (\sum_{i=1}^{n} h_i) \bmod q) \bmod q$ (predicted parity bits)and compare by $(\sum_{i=1}^{n} e_i) \bmod q$ (generated parity bits)if they are equal then there is no errors (except the case when the summation of errors is multiples of q.)

To show that

$$(\sum_{i=1}^{n} e_i) \bmod q = (p * (\sum_{i=1}^{n} r_i) \bmod q * (\sum_{i=1}^{n} h_i) \bmod q) \bmod q \qquad (7)$$

From Eq. 1 the party of *e(x)* will be

$$(\sum_{i=1}^{n} e(x)) \bmod q = (\sum_{i=1}^{n} (pr(x) * h(x) + m(x)) \bmod q) \bmod q \qquad (8)$$

Let

$$r(x) = r_0 + r_1\, x + r_2\, x^2 + r_3\, x^3 + \ldots\, r_n\, x^n$$

$$h(x) = h_0 + h_1\, x + h_2\, x^2 + h_3\, x^3 + \ldots\, h_n\, x^n$$

$$m(x) = m_0 + m_1\, x + m_2\, x^2 + m_3\, x^3 + \ldots\, m_n\, x^n$$

Substituting in (8)

$$e(x) = (pr(x) * h(x) + m(x)) \bmod q$$

$$\begin{aligned}
e(x) = {} & (p(r_0 + r_1 x + r_2 x^2 + r_3 x^3 + \ldots r_n x^n) * (h_0 + h_1 x + h_2 x^2 + h_3 x^3 + \ldots h_n x^n) \\
& + (m_0 + m_1 x + m_2 x^2 + m_3 x^3 + \ldots m_n x^n)) \bmod q \\
= {} & (pr_0 * (h_0 + h_1 x + h_2 x^2 + h_3 x^3 + \ldots h_n x^n) \\
& + pr_1 x * (h_0 + h_1 x + h_2 x^2 + h_3 x^3 + \ldots h_n x^n) \\
& + pr_2 x^2 * (h_0 + h_1 x + h_2 x^2 + h_3 x^3 + \ldots h_n x^n) \\
& + pr_3 x^3 * (h_0 + h_1 x + h_2 x^2 + h_3 x^3 + \ldots h_n x^n) \\
& + \ldots + pr_n x^n * (h_0 + h_1 x + h_2 x^2 + h_3 x^3 + \ldots h_n x^n) \\
& + (m_0 + m_1 x + m_2 x^2 + m_3 x^3 + \ldots m_n x^n)) \bmod q
\end{aligned}$$

$$(\sum_{i=1}^{n} e_i) \bmod q = (\sum coefficients(pr(x) * h(x) + m(x)) \bmod q) \bmod q$$

$$= (pr_0 * (h_0 + h_1 + h_2 + h_3 + \ldots h_n)$$

$$+ pr_1 * (h_0 + h_1 + h_2 + h_3 + ... h_n)$$

$$+ pr_2 * (h_0 + h_1 + h_2 + h_3 + ... h_n)$$

$$+ pr_3 * (h_0 + h_1 + h_2 + h_3 + ... h_n)$$

$$+ ... + pr_n * (h_0 + h_1 + h_2 + h_3 + ... h_n)$$

$$+ (m_0 + m_1 + m_2 + m_3 + ... m_n)) \bmod q$$

$$= (p(r_0 + r_1 + r_2 + r_3 + ... r_n) * (h_0 + h_1 + h_2 + h_3 + ... h_n) + (m_0 + m_1 + m_2 + m_3 + ... m_n)) \bmod q$$

$$= (p * (\sum_{i=1}^{n} r_i) * (\sum_{i=1}^{n} h_i) + (\sum_{i=1}^{n} m_i)) \bmod q$$

$$= (p * (\sum_{i=1}^{n} r_i) \bmod q * (\sum_{i=1}^{n} h_i) \bmod q + (\sum_{i=1}^{n} m_i) \bmod q) \bmod q$$

This is complete the proof of (8)

APPENDIX B

PARITY PREDICTION IN THE FIRST PART OF DECRYPTION PROCESS

In this section, we show the construction of the party prediction scheme of the first part of the decryption process. We will add parity bit for *a(x)* equal $(\sum_{i=1}^{n} a_i) \bmod q$ (predicted parity bits) then after the step of calculating *b(x)* we will compare the parity bit by $(\sum_{i=1}^{n} b_i) \bmod q$ (generated parity bits) if they are equal then there is no errors (except the case when the summation of errors is multiples of q.) We will use the following definition to proof that

$$(\sum_{i=1}^{n} a_i) \bmod q = (\sum_{i=1}^{n} b_i) \bmod q \qquad (9)$$

Definition. Let a(x) $\in Rq$. The *centered lift of* a(x) *to R* is the unique polynomial b(x) $\in R$ satisfying

$$b(x) \bmod q = a(x)$$

whose coefficients are chosen in the interval

$$-\frac{q}{2} < b_i(x) \le \frac{q}{2}$$

To show that $(\sum_{i=1}^{n} a_i) \bmod q$ equal $(\sum_{i=1}^{n} b_i) \bmod q$. From the previous definition $b_i(x) \bmod q = a_i(x)$

then

$$(\sum_{i=1}^{n} a_i) \bmod q = (\sum_{i=1}^{n} b_i(x) \bmod q) \bmod q$$

$$= (\sum_{i=1}^{n} b_i) \bmod q$$

This is complete the proof of (9)

APPENDIX C

PARITY PREDICTION IN THE SECOND PART OF DECRYPTION PROCESS

In this section, we show the construction of the party prediction scheme of the first part of the decryption process. We will add a parity bit for *b(x)* equal $(\sum_{i=1}^{n} b_i) \bmod p$ then after the step of calculating *m(x)* we will multiply the parity bit by $(\sum_{i=1}^{n} f_i) \bmod p$ and compare by $(\sum_{i=1}^{n} m_i) \bmod p$ if they are equal then there is no errors (except the case when the summation of errors is multiples of p.)

To show that

$$(\sum_{i=1}^{n} m_i) \bmod p = (\sum_{i=1}^{n} f_i) \bmod p * (\sum_{i=1}^{n} b_i) \bmod p \qquad (10)$$

Let

$$b(x) = b_0 + b_1 x + b_2 x^2 + b_3 x^3 + .. b_n x^n$$

$$f_p(x) = f_0 + f_1 x + f_2 x^2 + f_3 x^3 + ... f_n x^n$$

$$m(x) = (F_p(x) * b(x)) \bmod p$$

$$= ((f_0 + f_1 x + f_2 x^2 + f_3 x^3 + ... f_n x^n) * (b_0 + b_1 x + b_2 x^2 + b_3 x^3 + .. b_n x^n)) \bmod p$$

$$= (f_0 * (b_0 + b_1 x + b_2 x^2 + b_3 x^3 + .. b_n x^n)$$

$$+ f_1 x * (b_0 + b_1 x + b_2 x^2 + b_3 x^3 + .. b_n x^n)$$

$$+ f_2 x^2 * (b_0 + b_1 x + b_2 x^2 + b_3 x^3 + .. b_n x^n)$$

$$+ f_3 x^3 * (b_0 + b_1 x + b_2 x^2 + b_3 x^3 + .. b_n x^n)$$

$$+ ... f_n x^n * (b_0 + b_1 x + b_2 x^2 + b_3 x^3 + .. b_n x^n)) \bmod p$$

$$(\sum_{i=1}^{n} m_i) \bmod p = (\sum coefficients(f_p(x) * b(x))) \bmod p$$

$$= (f_0 * (b_0 + b_1 + b_2 + b_3 + ..b_n)$$

$$+ f_1 * (b_0 + b_1 + b_2 + b_3 + ..b_n)$$

$$+ f_2 * (b_0 + b_1 + b_2 + b_3 + ..b_n)$$

$$+ f_3 * (b_0 + b_1 + b_2 + b_3 + ..b_n)$$

$$+ ..f_n * (b_0 + b_1 + b_2 + b_3 + ..b_n)) \bmod p$$

$$= ((f_0 + f_1 + f_2 + f_3 + ...f_n) * (b_0 + b_1 + b_2 + b_3 + ..b_n)) \bmod p$$

$$= ((\sum_{i=1}^{n} f_i) * (\sum_{i=1}^{n} b_i)) \bmod p$$

$$= (\sum_{i=1}^{n} f_i) \bmod p * (\sum_{i=1}^{n} b_i) \bmod p$$

This is complete the proof of (10)

References

[1]Hoffstein J., Lieman D., Pipher J., Silverman J. "NTRU: A Public Key Cryptosystem", NTRU Cryptosystems, Inc. (www.ntru.com).

[2] Parasitism C, Prada J. "Evaluation of Performance Characteristics of Cryptosystem Using Text Files", Journal of Theoretical and Applied Information Technology, Jatit, 2008

[3] Jeffrey Hoffstein, Jill Pipher, Joseph H. Silverman"An Introduction to Mathematical Cryptography"2008

[4] G. Bertoni, L. Breveglieri, I. Koren, and V. Piuri, "Fault Detection in the Advanced Encryption Standard," Proc. Conf. MassivelyParallel Computing Systems (MPCS '02), pp. 92-97, 2002.

[5] AbdelAlim KAMAL *and* Amr YOUSSEF" Fault Analysis of the NTRUEncrypt Cryptosystem" IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences Vol.E94-A No.4 pp.1156-1158

[6] M. Akkar and C. Giraud, "Implementation of DES and AES, Secure against Some Attacks," Proc. Workshop Cryptographic Hardware and Embedded Systems (CHES '01), pp. 315-325, 2001.

[7] G. Bertoni, L. Breveglieri, I. Koren, P. Maistri, and V. Piuri, "Error Analysis and Detection Procedures for a Hardware Implementation of the Advanced Encryption Standard," Proc. IEEE TRANSACTIONS ON COMPUTERS, VOL.52, NO. 4, APRIL 2003.

[8] J. Hoffstein, D. Lieman, J. Pipher, J. H. Silverman, *NTRU: A Public Key Cryptosystem*, Submissions and Contributions to IEEE P1363.1, Presented at the August 1999 and October 1999 meetings. Avilable at http://grouper.ieee.org/groups/1363/lattPK/ submissions/ntru.pdf

[9] G. Bertoni, L. Breveglieri, I. Koren, P. Maistri, and V. Piuri, "On the Propagation of Faults and Their Detection in a Hardware Implementation of the Advanced Encryption Standard," Proc. Int'l Conf. Application-Specific Systems, Architectures, and Processors (ASAP '02), pp. 303-312, 2002.

[10] G. Bertoni, L. Breveglieri, I. Koren, P. Maistri, and V. Piuri, "A Parity Code Based Fault Detection for an Implementation of the Advanced Encryption Standard," Proc. IEEE Int'l Symp. Defect and Fault Tolerance in VLSI Systems (DFT '02), pp. 51-59, 2002.

[11] R. Karri, W. Kaijie, P. Mishra, and K. Yongkook, "Fault-Based Side-Channel Cryptanalysis Tolerant Rijndael Symmetric Block Cipher Architecture," Proc. Defect and Fault Tolerance in VLSI Systems (DFT '01), pp. 418-426, 2001.