İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

LOW COST NTRU IMPLEMENTATIONS

M.Sc. Thesis by
Ali Can ATICI

Department : Electronics and Communication Engineering

Programme : Electronics Engineering

JUNE 2009

İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

LOW COST NTRU IMPLEMENTATIONS

M.Sc. Thesis by
Ali Can ATICI
(504061203)

Date of submission : 04 May 2009
Date of defence examination: 03 June 2009

Supervisor (Chairman) : Ass. Prof. Dr. Berna ÖRS YALÇIN (ITU)
Members of the Examining Committee : Prof. Dr. Ece Olcay GÜNEŞ (ITU)
Prof. Dr. Ali Emre HARMANCI (ITU)

JUNE 2009

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

DÜŞÜK TÜKETİMLİ NTRU GERÇEKLEMELERİ

YÜKSEK LİSANS TEZİ
Ali Can ATICI
(504061203)

Tezin Enstitüye Verildiği Tarih : 04 Mayıs 2009
Tezin Savunulduğu Tarih : 03 Haziran 2009

Tez Danışmanı : Yrd. Doç. Dr. Berna ÖRS YALÇIN (İTÜ)
Diğer Jüri Üyeleri : Prof. Dr. Ece Olcay GÜNEŞ (İTÜ)
Prof. Dr. Ali Emre HARMANCI (İTÜ)

HAZİRAN 2009

FOREWORD

I would like to thank my supervisor Ass. Prof. S. Berna Örs Yalçın, for her support and guidance during my masters and encouraging me to go to Katholieke Universiteit Leuven. I also would like to thank my supervisors at Katholieke Universiteit Leuven, Dr. Lejla Batina and Prof. Ingrid Verbauwhede for their guidance and support during this thesis. I also would like to thank Junfeng Fan and Miroslav Knežević for their help during my thesis.

I would like to thank Abid Üveys Danış and Benedikt Gierlichs for their supervision and help in side channel analysis of my designs.

I would like to thank TUBITAK/BIDEB for their financial support during my masters.

Finally, I would like to thank my family for their endless support throughout my life.

June 2009

Ali Can ATICI
Electronics Engineer

TABLE OF CONTENTS

Page

ABBREVIATIONS

AES	:	Advanced Encryption Standard
CLA	:	Carry Lookahead Adder
CPA	:	Correlation Power Analysis
DES	:	Data Encryption Standard
DPA	:	Differential Power Analysis
ECC	:	Elliptic Curve Cryptography
FSM	:	Finite State Machine
LUT	:	Look-up Table
PM	:	Polynomial Multiplier
PRR	:	Partially Rotating Register
RFID	:	Radio Frequency Identification
SAIF	:	Switching Activity Interchange Format
SDF	:	Standard Delay Format
SPA	:	Simple Power Analysis
VCD	:	Value Change Dump
WSN	:	Wireless Sensor Networks

LIST OF TABLES

Page

LIST OF FIGURES

Page

LOW COST NTRU IMPLEMENTATIONS

SUMMARY

Radio Frequency Identification (RFID) tags and Sensor Nodes are emerging security platforms which are gaining popularity. RFIDs are already being used in supply chains, passports and it is envisioned that in the future they will be used in many new applications. Wireless Sensor Networks (WSN) can be used to monitor wildlife, the physical condition of bridges, airplane wings *etc.* Thus, the security of the information transferred by these devices is essential. To maintain this security, it is inevitable to use cryptographic algorithms in RFID and WSN systems. This work presents a compact and low power NTRU design that is suitable for pervasive security applications such as RFIDs and sensor nodes. NTRU is a public-key cryptosystem based on the shortest vector problem in lattices. Two different architectures have been designed for NTRU. One is only capable of encryption and the other one performs both encryption and decryption. To reduce the static power consumption, a low leakage library has been used. Furthermore, in order to reduce the dynamic power consumption, clock gating, operand isolation and precomputation methods are used wherever possible. This work is also the first one to present a complete NTRU design with encryption/decryption circuitry. Encryption-only NTRU design has a gate count of 2.8 kgates, dynamic power consumption of 1.72 μW and a latency of 56.44 ms at 500 kHz. Encryption-decryption NTRU design has a gate count of 10.5 kgates, 6 μW dynamic power consumption and a latency of 56.78 ms for encryption, 119.23 ms for decryption at 500 kHz. Moreover, to evaluate the resistance of the implementations against side channel attacks, Differential Power Analysis (DPA) attacks were conducted on the protected and unprotected implementations of the encryption-decryption NTRU design. The secret key in the implementations was inferred successfully with DPA attacks.

DÜŞÜK TÜKETİMLİ NTRU GERÇEKLEMELERİ

ÖZET

Radyo Frekansı ile Tanımlama (RFID) etiketleri ve Algılayıcılar, günümüzde popülarite kazanan ve güvenlik ihtiyaçları artan sistemlerdir. RFID etiketleri şimdiden tedarik zincirlerinde, pasaportlarda kullanılmaktadır ve ilerde başka alanlarda da kullanılması öngörülmektedir. Kablosuz Algılayıcı Ağları (WSN) vahşi yaşamı ya da bir köprünün, bir uçağın kanatlarının, *vb.* fiziksel durumunu gözlemlemede kullanılabilir. Dolayısıyla, bu cihazlarca aktarılan bilgilerin güvenliğinin sağlanması son derece önemlidir. Gerekli olan bu güvenliği sağlamanın yolu da şifreleme algoritmaları kullanmaktan geçmektedir. Bu çalışmada, RFID etiketleri ve algılayıcı ağları gibi sistemler için uygun olabilecek, az yer kaplayan ve düşük güç tüketimli NTRU gerçeklemeleri sunulmaktadır. NTRU, güvenliği kafeslerde en kısa vektör problemine dayanan, açık anahtarlı bir şifreleme algoritmasıdır. NTRU için biri sadece şifreleme, diğeri ise hem şifreleme hem şifre çözme işlemlerini yapabilen iki farklı devre tasarlanmıştır. Devrelerin statik güç harcamasını düşük tutmak için düşük kaçak akımlı teknoloji kütüphaneleri kullanılmıştır. Dinamik güç harcamasını azaltmak için ise saat işaretini engelleme, giriş izolasyonu ve önhesaplama yöntemlerinden faydalanılmıştır. Bu çalışma aynı zamanda, şifreleme/şifre çözme devrelerini bir arada bulunduran ilk NTRU tasarımıdır. Sadece şifreleme yapan NTRU devresi, 2.8 k kapılık bir alan kaplamış olup, 1.72 μW'lık dinamik güç harcaması vardır ve şifreleme işlemini f = 500 kHz için 56.44 ms'de bitirmektedir. Şifreleme/şifre çözme yapan NTRU devresi ise 10.5 k kapılık bir alan kaplamaktadır ve 6 μW'lık dinamik güç tüketimine sahiptir. F = 500 kHz için şifreleme 56.78 ms'de, şifre çözme ise 119.23 ms'de tamamlanmaktadır. Ayrıca, gerçeklemenin yan kanal analizlerine karşı olan dayanıklılığını ölçmek için, korumalı ve korumasız olarak gerçeklenmiş şifreleme/şifre çözme devresine farksal güç analizi (DPA) yöntemiyle saldırılmış ve sistemin gizli anahtarı elde edilmiştir.

1. INTRODUCTION

In communication systems, security of the transferred data has an essential importance, since the data mostly contains secret information. In order to maintain this security, cryptographic algorithms are widely used in software and hardware implementations. Radio Frequency Identification (RFID) tags and Wireless Sensor Networks (WSN) are new applications which have gained popularity as a result of their advantages in several fields.

RFID tags are wireless electronic devices which permit automated identification of objects and people [1]. RFID systems consist of three main components: transponder or RFID tag, transceiver or reader and database [2]. Each RFID tag has a microchip with some computation and storage capabilities and an antenna for communication. RFID tags can be passive, semi-passive and active depending on their power sources. RFID readers are devices that can interrogate the tags via RF communication. They also supply power for the passive tags through the RF signals. The information stored in the tags is mostly an index to a database which is reasonable when the restricted sources of the tags are considered.

RFIDs have a wide range of application fields. They can be used to improve the supply chain efficiency, to prevent the counterfeiting of medical drugs, to identify animals and objects, to store information for the health care services *etc.* [3]. Furthermore, RFID tags have already started to be used in toll payment systems, in libraries and in passports [1].

A sensor network node's hardware contains a microchip which is capable of some computing and storing, sensors, transceiver and a energy source [4]. They are able to sense the changes in moisture, temperature, vibration *etc.* around their environments. They are used to monitor the space, the things and the interaction of things with each other and their surroundings. WSNs can be used for environmental monitoring where humans can not make direct observations.

For example researchers use WSNs to monitor nesting seabird habitats. It is also possible to monitor the motion in physical structures like buildings, bridges, airplane wings and to obtain information about their conditions [4]. Then, this information can be used for maintenance and warning purposes.

In RFID and WSN systems, there are some informations which must be kept in secret, *i.e.* the personal informations in RFID tags. For example, the information regarding the condition of a bridge. Integrity and privacy of these informations must ensured against adversaries. This can be accomplished by the usage of cryptographic algorithms in these systems. However, implementing cryptographic algorithms in RFID tags and WSN is a challenging issue due to the restricted power and area resources. In RFID tags only 250-3000 gates can be allocated for cryptographic functions [2] and only about 20 μW power is available for the digital part of the passive tags [5]. In WSN systems, small batteries can supply up to 100 mW that is still not more than the necessity [5]. Thus, in order to ensure the security of the information in RFID and WSN systems, cryptographic algorithms must be implemented in a fashion to consume low power and low area.

The feasibility of Public-key (PK) solutions for RFIDs and WSNs is an open research problem due to severe limitations in costs, area and power. However, PKC protocols are useful for applications that need strong cryptography and services such as authentication, signatures, key-exchange *etc.* In addition, the use of PKC reduces power due to less protocol overhead [6].

1.1 Organization of the Thesis

In this dissertation, low cost implementations of the public-key algorithm NTRU [7] have been proposed which are viable for RFID and WSN systems. Chapter 2 gives brief information about the cryptosystems and explains the NTRU algorithm. The power dissipation sources in CMOS circuits and the low power design techniques among which some are used in the designs, are summarized in Chapter 3. Chapter 4 gives the implementation details of the encryption-only and encryption-decryption NTRU designs. Chapter 5 states the power, area and latency results of the designs and compares them with the previous works.

Chapter 6 explores the security of the designs against Differential Power Analysis (DPA) attacks. Finally, Chapter 7 concludes the thesis.

2. CRYPTOSYSTEMS AND NTRU ALGORITHM

Cryptography has been used for centuries in order to communicate securely, over unsecure channels. Especially nowadays, cryptography has the utmost importance to ensure the security of the digital data that is transferred, since people commonly send or receive private information over unsecure channels like Internet. There are two classes of cryptosystems that are used to maintain the data security: Symmetric-key and public-key cryptosystems.

2.1 Symmetric-key Cryptosystems

In symmetric-key systems both sides (sender and receiver) must share the same secret key K, which enables them to encrypt a message with the rule e_K and to decrypt a message with the rule d_K. In these systems e_K and d_K are same or slightly different [8, 9]. Figure 2.1 illustrates a symmetric-key cryptosystem.

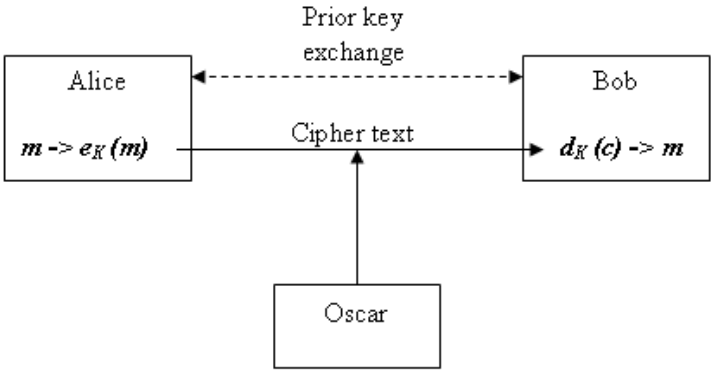


Figure 2.1: A symmetric-key cryptosystem

Here, Alice sends the message m to Bob by encrypting it with the key K. Thus, Oscar can see only the ciphertext which is meaningless to him. However, Bob who has the same key K will be able to decrypt the ciphertext and obtain the message m.

Advanced Encryption Standard (AES) [10] and Data Encryption Standard (DES) [11] are among the well known symmetric-key cryptosystems. A major

drawback of the symmetric-key systems is the problem of secret key sharing and management. The key must be distributed through a secure channel before communication, which is practically not so easy.

2.2 Public-key Cryptosystems

The idea of the public-key cryptosystems was first presented by Diffie and Hellman [12]. In these systems there exist a public key and a private (secret) key. The one who wants to send a message m uses the public key of the receiver to encrypt the message with the rule e_{pk}. Then, only the receiver who owns the secret key can decrypt the ciphertext with the rule d_{sk}. In public key cryptosystems, to derive the secret key from the public key must be computationally infeasible. Figure 2.2 shows a public-key cryptosystem.

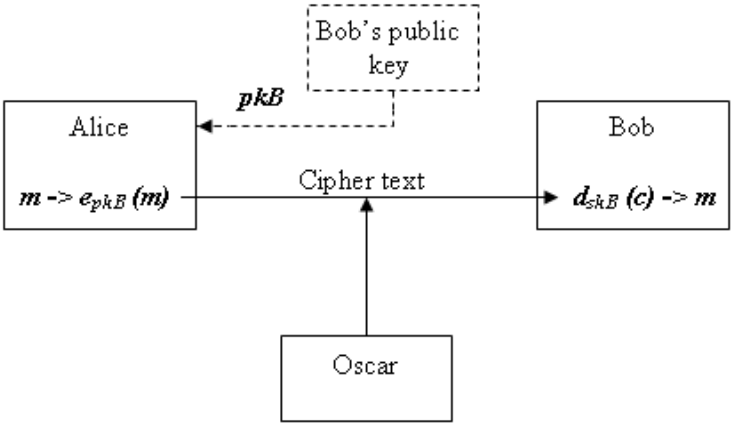


Figure 2.2: A public-key cryptosystem

When Alice wants to send a message to Bob, she encrypts the plaintext with Bob's public key p_{kB} and sends the ciphertext. Then, only Bob can decrypt the ciphertext with his secret key s_{kB}. The most popular public-key cryptosystems are RSA [13] and Elliptic Curve Cryptosystem (ECC) [14]. Since the encryption key is open in public-key cryptosystems, there is no need to deliver keys through a secure channel. Furthermore, public-key systems are also used in digital signature schemes [8, 9].

2.3 NTRU Algorithm

NTRU [7] is a public-key cryptosystem based on the shortest vector problem in a lattice. Basic operations of NTRU are realized in a truncated polynomial ring $R = Z[X]/(X^N - 1)$. Polynomials in the ring have integer coefficients and a degree of $N-1$. Addition in the truncated polynomial ring is carried out in a usual way simply by adding the coefficients that have the same degree [15].

$$a + b = (a_0 + b_0) + (a_1 + b_1)X + \ldots + (a_{N-1} + b_{N-1})X^{N-1} \tag{2.1}$$

However, multiplication is carried out in a slightly different way than the usual polynomial multiplication. During multiplication the rule $X^N \equiv 1$ is applied to all elements which have a degree equal or greater than N. This multiplication is called star multiplication [16] and denoted with the $*$ symbol. Thus, the product of two polynomials a and b

$$a(X) = a_0 + a_1X + a_2X^2 + \ldots + a_{N-1}X^{N-1} \tag{2.2}$$

$$b(X) = b_0 + b_1X + b_2X^2 + \ldots + b_{N-1}X^{N-1} \tag{2.3}$$

can be calculated as,

$$c(X) = a(X) * b(X) \tag{2.4}$$

$$c_k = a_0b_k + a_1b_{k-1} + \cdots + a_{N-1}b_{k+1} = \sum_{i+j \equiv k \bmod N} a_ib_j \tag{2.5}$$

In other notation if we think the polynomials a, b and c as coefficient vectors, then, the product $c = a * b$ simply equals to convolution product of two vectors as shown in Figure 2.3. There is no carry propagation between the columns.

	a_4	a_3	a_2	a_1	a_0
$\times$	b_4	b_3	b_2	b_1	b_0
	a_4b_0	a_3b_0	a_2b_0	a_1b_0	a_0b_0
	a_3b_1	a_2b_1	a_1b_1	a_0b_1	a_4b_1
	a_2b_2	a_1b_2	a_0b_2	a_4b_2	a_3b_2
	a_1b_3	a_0b_3	a_4b_3	a_3b_3	a_2b_3
$+$	a_0b_4	a_4b_4	a_3b_4	a_2b_4	a_1b_4
	c_4	c_3	c_2	c_1	c_0

Figure 2.3: Convolution product of two vectors

NTRU public-key cryptosystem has three integer parameters (N, q, p) and four sets L_f, L_g, L_r, L_m of polynomials of degree $N-1$ which have all integer coefficients. It is assumed that N is prime, $gcd(p, q) = 1$ and q is always fairly larger than p.

Security level of NTRU depends on the selection of these parameter sets [7]. Table 2.1 shows the security levels of NTRU according to the chosen parameter sets. Furthermore, NTRU with the parameter sets $(N,q,p) = (251,128,3)$ and

Table 2.1: NTRU security levels

	N	q	p
Moderate Security	107	64	3
High Security	167	128	3
Higest Security	503	256	3

$(N,q,p) = (503,256,3)$, respectively has an equivalent level of security with 1024-bit RSA, 163-bit ECC and 4096-bit RSA, 320-bit ECC [16, 17].

2.3.1 Key generation

To generate key sets of NTRU, one must first choose two random small polynomials $f \in L_f$ and $g \in L_g$. In this context small polynomial refers to a polynomial whose coefficients are much smaller than q. The polynomials f and g must be kept in private to prevent others obtain the private keys. In the next step, the inverse of polynomial f must be computed in modulo q and modulo p. So, $f_q \equiv f^{-1} \pmod{\text{q}}$ and $f_p \equiv f^{-1} \pmod{\text{p}}$ must exist, otherwise another f must be selected for the key generation process. As the last step, h is calculated:

$$h \equiv f_q * g \pmod{\text{q}} \tag{2.6}$$

Now, the private keys of the system are f and f_p and the public key of the system is h. For more information about parameter selection, small polynomials and finding inverses of polynomials can be found in [7, 18, 19].

2.3.2 Encryption

Firstly, a message m must be chosen from the set of plain texts L_m. m must be a small polynomial whose coefficients practically lie between $\lceil -p/2 \rceil$ and $\lfloor p/2 \rfloor$. Then, a random small polynomial r must be selected from the set L_r, to use as blinding value. Finally, encrypted text can be evaluated as follows:

$$e \equiv pr * h + m \pmod{\text{q}} \tag{2.7}$$

In the encryption process, h is always multiplied with p. Since these values do not change for the following encryptions, it is suggested to use the public key $h \equiv pf_q * g \pmod{q}$ instead of $h \equiv f_q * g \pmod{q}$ [15].

2.3.3 Decryption

In order to decrypt a received message e, one must first compute

$$a = f * e \pmod{q} \tag{2.8a}$$
$$= f * (r * h + m) \pmod{q} \tag{2.8b}$$
$$= f * (r * pf_q * g + m) \pmod{q} \tag{2.8c}$$
$$= pr * g + f * m \pmod{q} \tag{2.8d}$$

At this stage it is essential to chose the coefficients of a between $-q/2$ and $q/2$ instead of 0 and $q-1$. Otherwise, message may not be properly recovered. After this step,

$$b = a \pmod{p} \tag{2.9a}$$
$$= f * m \tag{2.9b}$$

should be calculated. And, as the final step, plaintext is recovered with the following calculations:

$$c = b * f_p \pmod{p} \tag{2.10a}$$
$$= f_p * f * m \pmod{p} \tag{2.10b}$$
$$= m \tag{2.10c}$$

3. LOW POWER DESIGN TECHNIQUES

The main concern in designs which have restricted resources, is to reduce the power dissipation, as much as possible. There exists many methods in the literature which are devoted to this subject. These power reduction methods can be classified into two, as technological level methods and architectural level methods. These methods are briefly mentioned in this section. The details of these methods can be found in [20–23].

3.1 Sources of Power Dissipation

In CMOS digital circuitry there are two kinds of power dissipation [20]: Dynamic and static. For clarity, the causes of dynamic and static power dissipation is explained on a CMOS inverter.

3.1.1 Dynamic power dissipation

Dynamic power is the power consumed by a gate when that gate is active which means there is a switching activity in the logic levels of the gate. The major source of the dynamic power dissipation is the switching power which is consumed to charge and discharge the output capacitance of the gate [22].

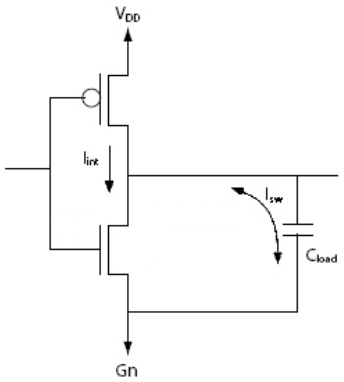


Figure 3.1: Dynamic power dissipation

In case of a '1'→'0' transition at the input of the inverter, the capacitance will be charged by the pMOS transistor and for a '0'→'1' transition, the capacitance will be discharged by the nMOS transistor. C_{load} is the total capacitance from transistors and interconnect wires and V_{DD} is the supply voltage. The energy for one transition is:

$$Energy/transitions = C_{load} \times V_{DD}^2 \tag{3.1}$$

Then, switching power can be formulated as:

$$P_{dyn} = Energy/transitions \times f = C_{load} \times V_{DD}^2 \times P_{trans} \times f_{clk} \tag{3.2}$$

f is the frequency of transitions, P_{trans} is the probability of an transition and f_{clk} is the system clock. If we put $C_{eff} = P_{trans} \times C_{load}$, then it is also possible to express the dynamic power as:

$$P_{dyn} = C_{eff} \times V_{DD}^2 \times f_{clk} \tag{3.3}$$

As it can be seen dynamic power does not depend on transistor parameters, but depends on the switching activity and load capacitances. Hence, it is data dependent.

Another source of dynamic power consumption is the internal power. During the transition of the gate there happens a time that both transistors are on. At that period a short circuit current flows from V_{DD} to ground and causes a internal power dissipation. But with careful design of the circuits internal power can be kept below the 10-15 % of the total power consumption [21]. This component can be expressed in the dynamic power equation as follows:

$$P_{dyn} = (C_{eff} \times V_{DD}^2 \times f_{clk}) + (t_{sc} \times V_{DD} \times I_{peak} \times f_{clk}) \tag{3.4}$$

Where t_{sc} is the duration of the short circuit current and I_{peak} is the total internal current (short circuit current plus the current needed to charge the internal capacitance).

3.1.2 Static power dissipation

Static power is the power dissipated by a gate when that gate is inactive. The two main source of the static power dissipation in CMOS circuits are subthreshold

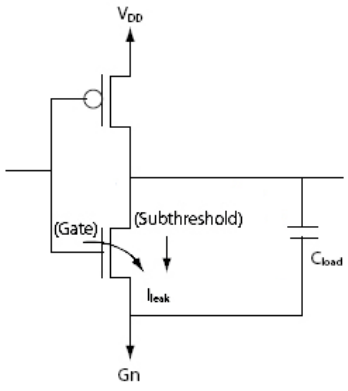


Figure 3.2: Static power dissipation

leakage currents and gate leakage currents [21, 22]. Subthreshold leakage is the current that flows from the drain to the source due to reduced threshold voltages that prevent the gate from completely turning off. Gate leakage is the current which flows from the gate through the oxide to the substrate. It is possible to formulate the static power as in the Equation 3.5.

$$P_{sta} = I_{leak_{Tot}} \times V_{DD} \tag{3.5}$$

Leakage currents are increasing, as the CMOS technology improves, due to low threshold voltages. It is known that in the future, static power dissipation will play a major role like the dynamic power dissipation [20, 22].

3.2 Technological Level Power Reduction Methods

Some well known technological level power optimization methods are explained under this section, although the only method applied in this work is the usage of a low leakage technology library. However these methods can be used in order to obtain better results.

3.2.1 Multi V_{DD}

Lowering the supply voltage V_{DD} decreases the dynamic power, since the power dissipation is proportional to V_{DD}^2. But it also decreases the speed of the gates. Thus, the idea is to use different supply voltages for the different modules of the system, still with meeting the system timing constraints. In Figure 3.3 a multi V_{DD} system is illustrated. CPU and Cache RAMs are supplied with high voltage to meet the system timing while the rest of the chip is supplied with low voltage.

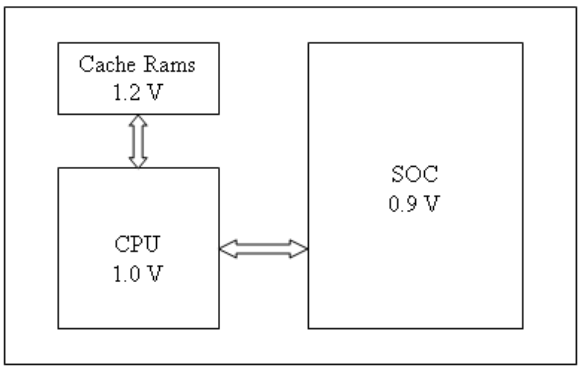


Figure 3.3: Multi V_{DD} system

Hence, each component of the system is working in the lowest possible voltage with meeting the timing requirements, to save power.

3.2.2 Multi-threshold logic

Technology libraries can contain more than one version of the same cell with different threshold voltages. By the help of the implementation tools this property can be used to reduce the static power dissipation. The method is to use fast, low V_T and leaky transistors only on the critical timing paths to meet the timing and for the rest of the design to use slower, high V_T and less leaky transistors.

3.2.3 Power gating

In this method the aim is temporarily to turn off the blocks that are not being used and reducing the static power dissipation. One way to realize this idea is using an externally switched power supply. A unit that has an off-chip power supply can be turned off by shutting down its power supply and the leakage in that unit can be reduced to zero. However, this method takes the longest time and requires the most energy to restore power to a gated block.

Another approach for power gating is to use internal switches to disconnect the gates from V_{DD} and/or ground. In practical, disconnecting the gate from one of V_{DD} or ground is enough. These power switches can be implemented in each cell separately. But, a more area efficient method of implementing these switches is to use one switch for a block of gates as shown in Figure 3.4.

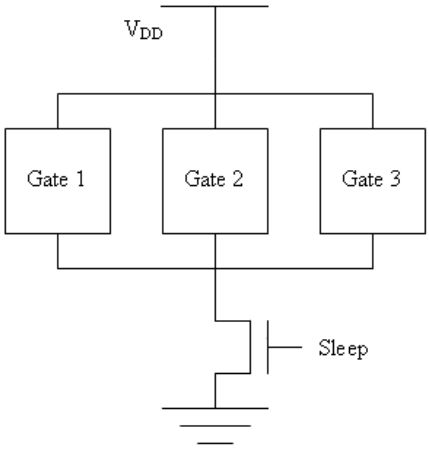


Figure 3.4: Power gated blocks

3.3 Architectural Level Power Reduction Methods

In this work, mainly architectural level optimizations, such as clock gating, gate level optimizations, operand isolation and precomputation, provide power reduction. These methods were used wherever possible in the designs.

3.3.1 Clock gating

Clock gating is an effective method to decrease the dynamic power consumption. The aim is to disable the clock of a component when its outputs are not used. Thus, switching activity of the flip-flops, gates in the fan-out of the flip-flops and the clock trees can be reduced. The most preferred clock gating circuitry contains a latch to prevent glitches occurring. Figure 3.5 shows a register bank implemented in a typical fashion. In this implementation, registers switch every

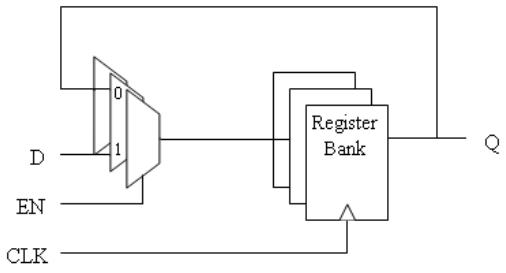


Figure 3.5: Typical register implementation

clock cycle even they are not selected by the enable signal. So, to prevent this unwanted switching activity, registers can be implemented with a clock gating circuitry as illustrated in Figure 3.6. With this implementation registers switch

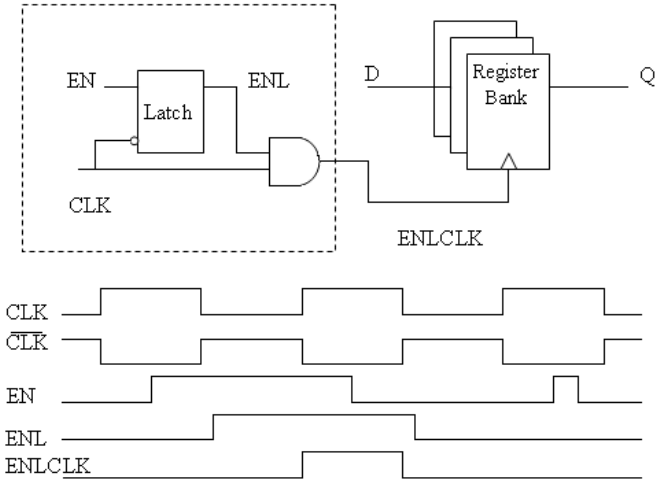


Figure 3.6: Clock gated register implementation

only when their output is selected. Note also that, clock gating saves circuit area as a result of the removal of the many MUXs with one clock gating circuitry.

3.3.2 Gate level optimizations

The approach in this method is remapping the logic gates of the components to obtain less power dissipating structures without changing their functions. Above of the Figure 3.7, the output of the AND gate is remapped to make the high activity net internal to the cell. The output of the AND gate is now driving a smaller capacitance node. Thus, dynamic power is reduced.

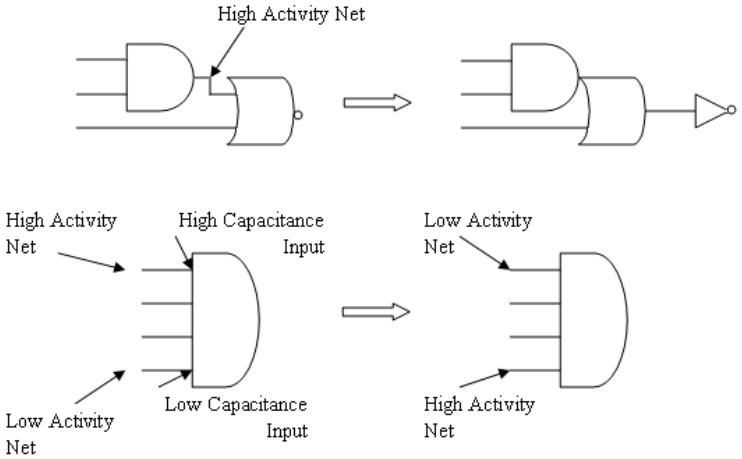


Figure 3.7: Gate level optimizations

Below of the Figure 3.7, a four input AND gate is firstly mapped as, a high activity net is connected to a high capacitance pin and a low activity net is connected to a low capacitance pin. By swapping these inputs so that high

activity net is connected to a low capacitance pin, dynamic power consumption can be decreased.

3.3.3 Operand isolation

The aim of the operand isolation is to isolate the inputs of the complex combinational logic gates such as adders and multipliers whenever their outputs are not selected. Thus, isolation logic prevents the gates switching unnecessarily and reduces dynamic power consumption.

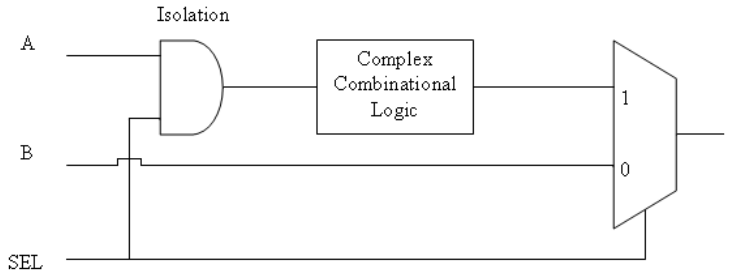


Figure 3.8: Operand Isolation

As illustrated in Figure 3.8 input A is allowed to enter the complex logic only if the output of thecomplex logic is selected.

3.3.4 Precomputation

This method is based on the fact that in many cases there are some input values for which the output of the circuit can be computed easily. That is to say, the input values of the system are checked and for some values output is computed by an other circuit which is smaller than the main circuit. This allows reducing the dynamic power in the main circuit.

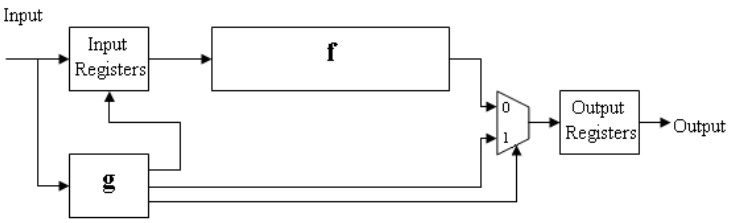


Figure 3.9: A precomputed circuit architecture

In the Figure 3.9, the input values are checked by **g**. For some values the input registers are disabled. So, the switching activity of **f** is stopped. The result

is computed by **g** and given to the output through multiplexer. However, this method also brings increase in the delay and the area of the circuit due to the added extra logic.

4. NTRU IMPLEMENTATION

In this work, two NTRU implementations has been proposed. One design is only capable of encryption and the other one is capable of both encryption and decryption. Designs were written in GEZEL [24] hardware design language. For the designs, a parameter set $(N,q,p) = (167,128,3)$ was used which yields to a high level of security [7]. Moreover, during the design phase, power consumption reduction methods, such as clock gating, precomputation and operand isolation, were used wherever possible.

4.1 Implementation of Encryption-Only NTRU

In the encryption-only NTRU design, as a result of the selected parameter set $(N,q,p) = (167,128,3)$, public key h has coefficients from the interval $[0,127]$, random polynomial r and message m have coefficients from the set $\{-1,0,1\}$. Thus, the coefficients of h are encoded in 7-bits and the coefficients of m and r are encoded in 2-bits. While dealing with the negative numbers, two's complement representation is used.

The main architecture of the encryption-only design, as shown in Figure 4.1, consists of a look-up table (LUT), a polynomial multiplier (PM), a partially rotating register (PRR) and a control logic. In the design, *enc* input is used to detect the start signal, r_{in} and *message* inputs are used to receive random polynomial and message respectively. Output *done* is used to inform the ending of the computation of a new result and *result* output is used to give the encrypted text coefficients.

Random polynomial generation is beyond the scope of this work, so it is assumed that circuit has an outer source for random number generation, in order to produce the random polynomial r.

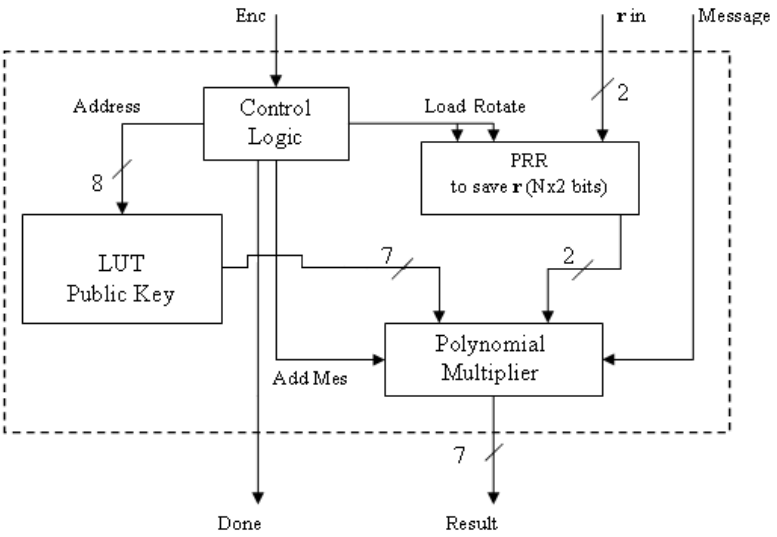


Figure 4.1: Encryption-only NTRU architecture

4.1.1 Look-up table

In order to reduce the computation load, public key $h \equiv pf_q * g \pmod q$ is precomputed and stored as a look-up table. The LUT consists of only combinational circuit. It gives the 7-bit public key coefficient according to the value of its 8-bit address input.

4.1.2 Polynomial multiplier

The polynomial multiplier (PM) is the arithmetic core of the design where all computations are carried out. It has a 7-bit multiplier, a 7-bit Carry Lookahead Adder (CLA), and a 7-bit register as shown in Figure 4.2. All of the submodules of the PM are realized for 7-bit computations since all operations are done modulo 128. This way, intermediate and final results are directly reduced in modulo 128 during computations, simply by ignoring the 8^{th} bit.

Due to the small coefficients of random polynomial r, multiplier has a simple structure. Multiplier has one 7-bit *multiplicand* input and one 2-bit *multiplier* input. Since the coefficients of r take values only from the set $\{-1,0,1\}$, the product of a multiplicand x is easily computed from the set $\{-x,0,x\}$. In the case of a negative multiplier, the product is generated by inverting the bits of the multiplicand and setting the carry input of the following CLA adder to '1'.

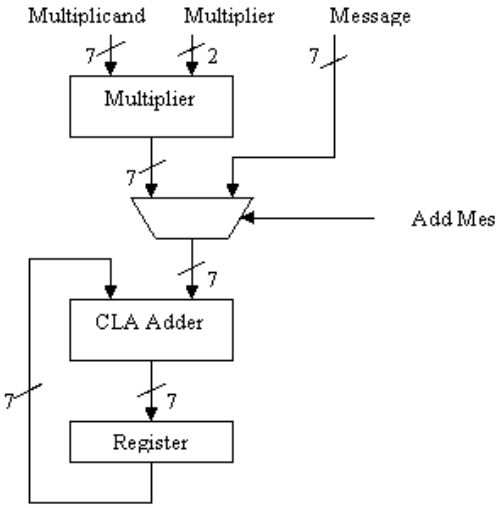


Figure 4.2: Polynomial multiplier

CLA adder is selected for the implementation because of its non-carry propagation property which is needed for less switching activity [25, 26]. Also CLA adder is implemented in two blocks which are 4-bits and 3-bits length. Intermediate products are saved in the 7-bit register and the value of the register is always fed back to the CLA adder input. The value of the register is updated by the sum of the partial product and the current register value. After all multiplication and accumulation steps, *add mes* input is enabled and message is added to the current value of the register to generate the final result. To reduce the switching activity at the output of the PM, *result* output remains unchanged until final results are computed.

For a more clear view of the encryption multiplication, one may think h and r as the polynomials b and a in Figure 2.3 respectively.

4.1.3 Partially rotating register

Partially rotating register (PRR) is used to store and rotate the random polynomial r's coefficients and it is $N \times 2$ bits wide. Before encryption all coefficients of r are loaded into PRR and then encryption starts. During encryption, the random polynomial coefficient at the input of PM changes for every public key coefficient, which means, for every partial product, r's coefficient value must be updated. So in case of a regular rotating register, all of the bits must be shifted once, which means switching activity of $N \times 2$ registers for every partial product computation. Simulations have shown that clock gating a regular

rotating register, does not have a positive effect on the power consumption in the case of NTRU encryption, due to the continuous rotation of the register. On the contrary, it increases the dynamic power consumption owing to the extra switching activity that comes from the gating circuits (See Table 5.1). To overcome this negative situation and to benefit from clock gating maximally, a partially rotating register was designed, as illustrated in Figure 4.3.

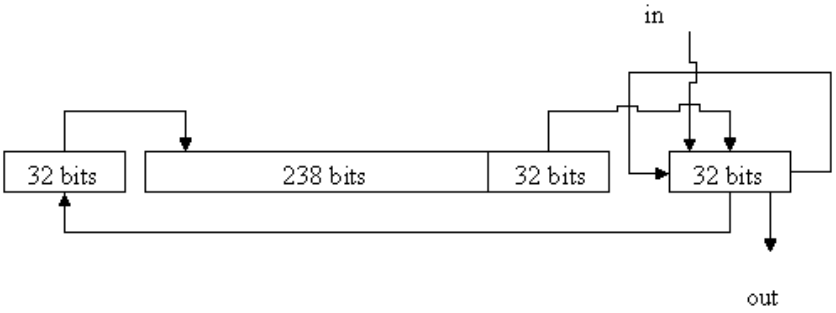


Figure 4.3: Partially rotating register (PRR)

In this architecture the right hand side 32-bit register is the part that always rotates during encryption and loading of the random polynomial r. The two most significant bits of that register are used as data input during loading phase. Output of the PRR is always the least significant two bits of that register. A *partial rotate* signal is sent by the controller which only enables rotating of the right hand side 32-bit register. After all 16 values of that register are used in the computations, a *whole rotate* signal is generated to renew the values of the register. This signal causes the PRR to make a block rotation with width of 32-bits. With this method, only 32 bits of the PRR switch constantly while the rest of the registers switch only 9 times in the computation of each cipher text word and during loading of r.

4.1.4 Control logic

Controller of the encryption engine was designed as a finite state machine (FSM) which has four states. It controls the whole process with two 8-bit counters and one 4-bit counter. One of the 8-bit counters is used to count the number of intermediate products in one cipher text computation, while the other one is used to count the number of the produced cipher text coefficients. The 4-bit counter is used to control the *partial rotate* and the *whole rotate* signals of the PRR.

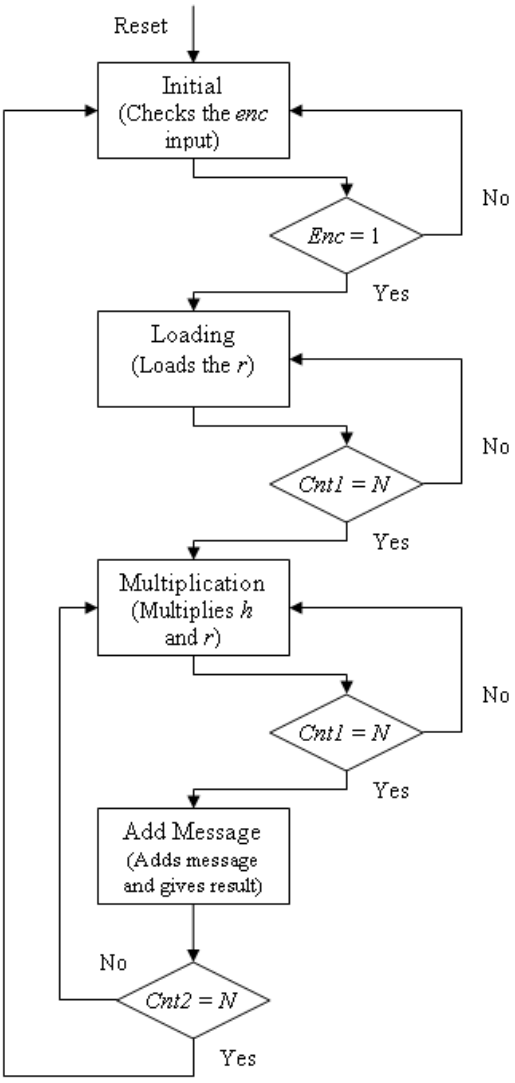


Figure 4.4: FSM of encryption-only NTRU

Figure 4.4 shows the state transition diagram of the FSM. Here *Cnt1* and *Cnt2* refer to 8-bit counters. Circuit produces the encrypted text polynomial coefficients from the least significant one to the most significant one. The FSM works as follows:

- *Initial:* It is the start state of the circuit after reset signal. In this state circuit only checks its *enc* input and if detects a high signal at the input, transits to the *loading* state.

- *Loading:* During this state, the coefficients of the random polynomial r are loaded into the PRR one by one. After loading all of the coefficients (N clock cycles), FSM passes to the *multiplication* state.

- *Multiplication* : This is the state which the multiplication $h * r$ is carried out. The coefficients of h and r are multiplied and accumulated in this state. After finishing all the N multiplications and accumulations (N clock cycles) that is needed for the generation of one cipher text coefficient, FSM goes to the *add message* state.

- *Add Message:* At this last state, message is added to the current sum value and *done* output is made '1' for one clock cycle (1 clock cycle). Unless, it is the last coefficient, FSM transits to the *multiplication* state to calculate the next cipher text coefficient, otherwise it transits to the *initial* state.

4.2 Implementation of Encryption-Decryption NTRU

In the encryption-decryption NTRU, the same parameter set was used. Thus, again the coefficients of h and r are encoded in 7-bits and 2-bits respectively. Also, in this case two more polynomials that are private keys f and f_p was used. f has coefficients from the set $\{-1,0,1\}$ and f_p has coefficients from the set $\{0,1,2\}$. Hence, the coefficients of f and f_p are also encoded in 2-bits. Figure 4.5 illustrates the architecture of encryption-decryption NTRU. For simplicity of the figure, not all of the modules and inner signals are shown.

There are extra modules in the design which have been added to the encryption-only NTRU architecture. These are, two more look-up tables to store the private keys f and f_p, a Mod-3 unit, a $N \times 7$ bits wide PRR and a result register. Moreover, there are four routers, composed by multiplexers, which direct the correct values to the correct inputs.

Look-up tables give the key coefficient value according to their 8-bit address input. LUT of h has a 7-bit output whereas LUT's of f and f_p have 2-bit outputs.

4.2.1 Polynomial multiplier

The architecture of the PM is the same as in Figure 4.2, but some modifications have been done to make the circuit, decryption compatible. Since, f_p is now included in multiplication, some extra circuitry was added to the multiplier so it is capable of multiplying by 2. Multiplication by 2 is simply achieved by

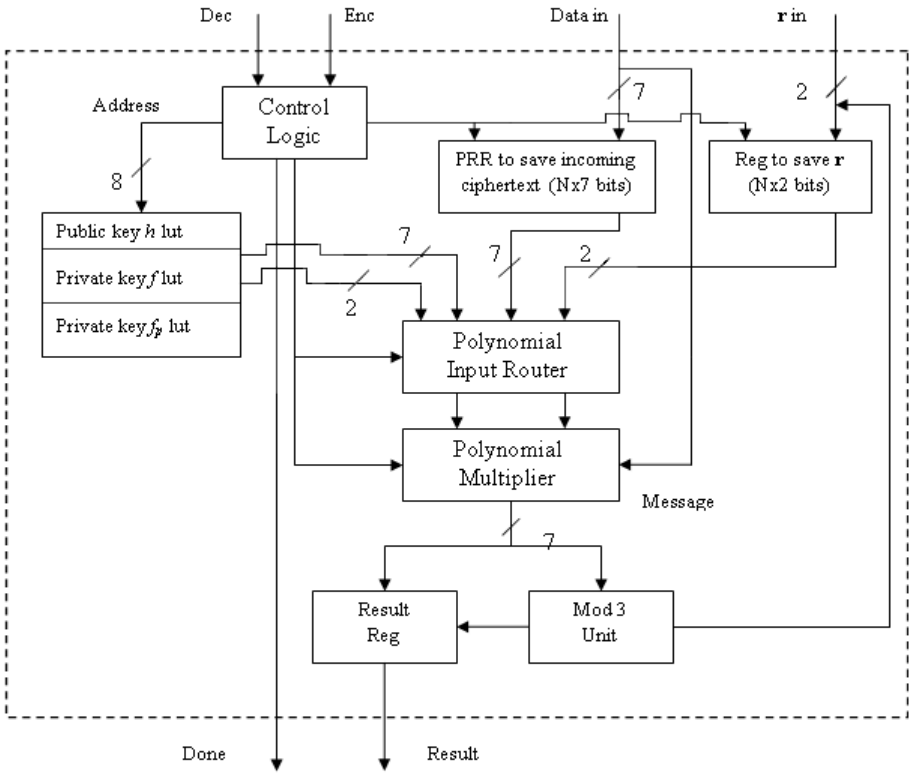


Figure 4.5: Encryption-decryption NTRU architecture

shifting the multiplicand to the left once. Reduction modulo 128 is performed by ignoring the most significant bit of the shifted multiplicand and inserting a '0' to the least significant bit of the shifted value. Table 4.1 shows the truth table of the multiplier.

Furthermore, an *overflow* output was added to the PM, which is the carry output of the CLA adder, to count the number of modulo 128 reductions in the second multiplication of decryption. Since, the PM is designed to work only in modulo 128, the number of modulo 128 reductions is used to obtain the correct results, where the multiplications must be carried out in modulo 3.

Table 4.1: Truth table of the PM multiplier

Multiplier		Multiplicand	Output
Bit representation	Decimal value		
00	0	$m_6m_5m_4m_3m_2m_1m_0$	0000000
01	1	$m_6m_5m_4m_3m_2m_1m_0$	$m_6m_5m_4m_3m_2m_1m_0$
10	2	$m_6m_5m_4m_3m_2m_1m_0$	$m_5m_4m_3m_2m_1m_00$
11	-1	$m_6m_5m_4m_3m_2m_1m_0$	$m'_6m'_5m'_4m'_3m'_2m'_1m'_0$

4.2.2 Rotating registers

There are two rotating registers in the design. One is a $N \times 7$ bits PRR and the other one is a $N \times 2$ bits regular rotating register. PRR is used to store and rotate the incoming cipher text. Regular rotating register has two functions. Its first function is to store and rotate the random polynomial r during encryption and the second one is to store and rotate the intermediate result $b = a \pmod{p}$ [$a = f * e \pmod{q}$] which occurs during decryption. $N \times 7$ bits wide PRR has the same architecture as the one in Figure 4.3, only with modified bit lengths. It also rotates in the same direction. However, register $N \times 2$, rotates in the reverse direction, as shown in Figure 4.6.

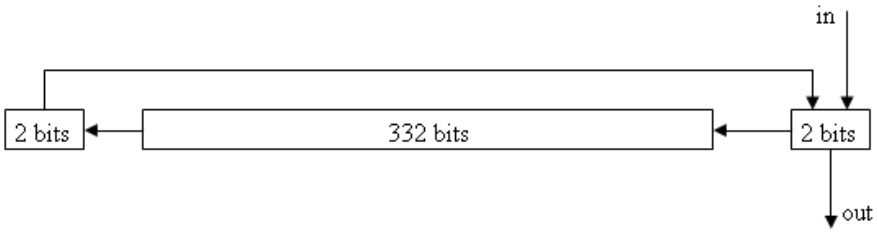


Figure 4.6: Rotating register $N \times 2$

The least significant two bits are used as inputs during loading of the register and these two bits are also the output of the register.

4.2.3 Mod-3 unit

There are two reduction 3 operations during decryption. The First one is $b = a \pmod{p}$ and the second one is $c = b * f_p \pmod{p}$. These modulo 3 reductions are performed by a finite state machine based circuit [27]. The state transition diagram of the FSM is illustrated in Figure 4.7.

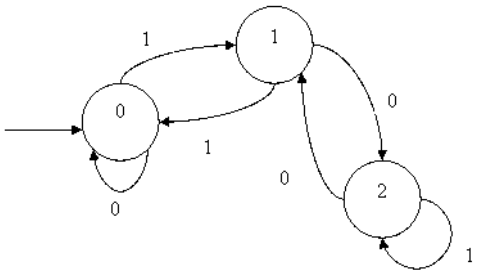


Figure 4.7: Mod-3 FSM

The FSM starts with the state 0 and checks the bits of the binary number from the most significant bit to the least significant bit. According to the value of the bit that is being checked, it makes state transitions. The final state after checking the last bit, equals to the modulus 3 value of that number.

Furthermore, *overflow* output of the PM is connected to this module. There is a 2-bit overflow counter which detects and counts the overflows that occurs in the PM. The counter always starts with 0 to a new reduction and counts in the sequence $\{0,2,1,0,2,1,\ldots\}$ which denotes the overflow sums in modulus 3 as shown in Figure 4.8.

1^{st} Overflow	$128 \equiv 2$	mod (3)
2^{nd} Overflow	$256 \equiv 1$	mod (3)
3^{rd} Overflow	$384 \equiv 0$	mod (3)
4^{th} Overflow	$512 \equiv 2$	mod (3)
$\vdots$		$\vdots$

Figure 4.8: The overflow sums in modulus 3

The correct modulus 3 result of the input is calculated by concatenating the value of the counter to the end of the input bits and checking that value by the module's FSM. In the first reduction by 3, results are generated from the set $\{0,1,2\}$ to decide the number of overflows in the following (2^{nd}) multiplication of decryption correctly, whereas in the second reduction 3, results are generated from the set $\{-1,0,1\}$.

The coefficients of the first decryption multiplication $a = f * e \pmod{q}$ must lie between $-q/2$ and $q/2$ before performing reduction 3. This operation is also performed in Mod-3 unit. In order to place the coefficients in the interval $[-q/2, q/2]$, q must be subtracted from the coefficients that are bigger than $q/2$. Thus, negative numbers involve in reduction by 3. But in this design, instead of dealing with negative numbers, another way has been chosen. If $x > 64$, instead of calculating $(x-128) \equiv y \pmod 3$, $\{(x \equiv a \pmod 3) - (128 \equiv 2 \pmod 3)\} \equiv y \pmod 3$ is computed. The circuit first computes the modulo 3 value of x which is a and then gives to the output, 2 smaller of a as the result.

4.2.4 Routers

The four routers in the architecture are, LUT input address router, PM input router, rotating registers input router and result register input router. Under the supervision of the control logic, these routers provide three different connection configurations for the architecture. These configurations are denoted as, *Enc, Dec1, Dec2.*

- *Enc:* This is the configuration needed for encryption. Under this configuration, 7-bit and 2-bit inputs of the PM are bounded to LUT h and register $N \times 2$ outputs respectively. The message input of PM is connected to *data in*. Input of the register $N \times 2$ is connected to *r in* and output of PM is connected to result register input.

- *Dec1: Dec1* is the configuration of the first multiplication and modulo 3 reduction during decryption process. Under *Dec1*, *data in* is connected to the input of the PRR. 7-bit and 2-bit inputs of the PM are bounded to PRR and LUT f outputs respectively. Output of the PM is connected to the input of Mod-3 unit, while the output of the Mod-3 unit is connected to the input of register $N \times 2$.

- *Dec2:* The third configuration *Dec2* belongs to the second multiplication and modulo 3 reduction of decryption. Here, 7-bit and 2-bit inputs of the PM are connected to LUT f_p and register $N \times 2$ respectively. Output of PM is again bounded to Mod-3 unit input but the output of the Mod-3 unit is connected to the result register.

4.2.5 Control logic

Controller of the encryption-decryption NTRU is also designed as a finite state machine which has seven states. It controls the encryption and decryption processes with two 8-bit counters and one 4-bit counter. Again, similar to the encryption-only NTRU controller, 8-bit counters are used to control the nested multiplication loops, while 4-bit counter is used to control PRR. Figure 4.9 illustrates the state transition diagram of the design.

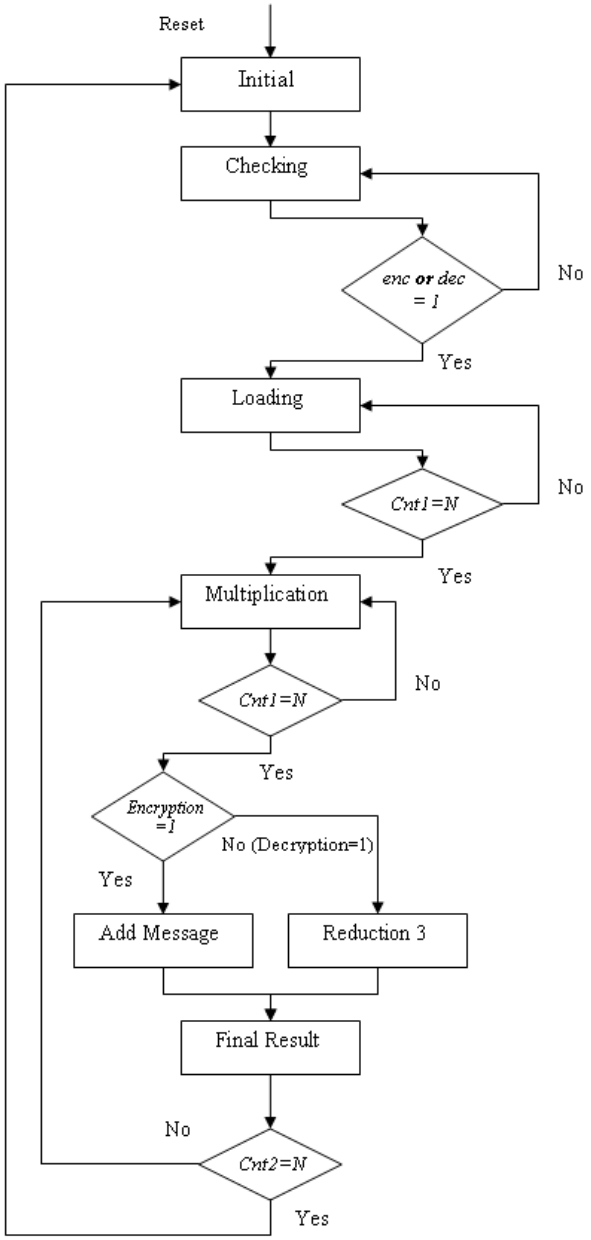


Figure 4.9: FSM of encryption-decryption NTRU

Cnt1 and *Cnt2* refer to the 8 bit counters. Circuit generates the cipher text coefficients from the least significant one to the most significant one. Also during decryption, it receives the cipher text coefficients in the same order and produces the plain text again from the least significant coefficient to the most significant coefficient. The FSM works as follows:

- *Initial:* The FSM starts with this state after reset and goes directly to the *checking* state with controlling the *enc* and *dec* inputs.

- *Checking:* At this state controller checks the *enc* and *dec* inputs every clock cycle. On detection of a high signal at one of these inputs, FSM transits to the *loading* state, otherwise stays at this state.

- *Loading:* If it is encryption, coefficients of r are loaded to the register $N \times 2$ one by one (N clock cycles), else if it is decryption, coefficients of cipher text e are loaded to the PRR one by one (N clock cycles). *Loading* state is followed by the *multiplication* state.

- *Multiplication: Multiplication* state is common for encryption and decryption since polynomial multiplication is carried out in the same way in both situations (N clock cycles). However, the next state depends on the computation. If it is encryption the FSM transits to the *add message* state, otherwise it transits to the *reduction 3* state.

- *Add Message:* At this state message is added to the current sum (1 clock cycle). *Add Message* is followed by the *final result* state.

- *Reduction 3:* At *reduction 3* state, the result generated by PM is loaded in Mod-3 unit and the modulo 3 value of that number is calculated. Reduction takes $n+3$ clock cycles where n is the bit length of the input value. *Reduction 3* state is also followed by the *final result* state.

- *Final Result:* If the circuit is performing an encryption, controller loads the result register (1 clock cycle) with the current result which was generated after the *add message* state and makes the *done* output '1' for one clock cycle. If decryption is being carried out and the circuit is performing the first multiplication, the FSM loads the reduction 3 result to the register $N \times 2$ (1 clock cycle). If the second multiplication is being carried out, then reduction by 3 result is loaded into the result register (1 clock cycle) and *done* output is made '1' for one clock cycle. For all situations, if the last coefficient is given out, system transits to the *initial* state, otherwise it transits back to *multiplication* state.

5. RESULTS AND ANALYSIS

The NTRU designs were written in GEZEL [24] hardware design language and optimized for low power and low area. Since the circuit is planned to be running at low frequencies, a low leakage technology library was used for synthesis process. As the technology library, the Faraday Low Leakage $0.13\,\mu m$ library was selected. For the synthesis, the VHDL codes, which were generated automatically by GEZEL, were used. Designs were synthesized by Synopsys Design Vision [28] at 500 kHz. RTL level and gate level simulations were done by ModelSim. All power estimations were done by Synopsys Power Compiler [23] and these measurements were carried out by the switching activity which was captured by the gate level simulations in ModelSim.

5.1 Power Estimation Methodology

In order to estimate the power consumption of the designs as accurate as possible, power simulations were done in the gate-level by using the switching activity. Figure 5.1 shows the power estimation flow.

After compiling the design with Synopsys, a verilog netlist and a standard delay format (SDF) file are produced in order to be used in simulations. The SDF file contains the timing information of the cells in the netlist. After this step the netlist is simulated in ModelSim and a value change dump (VCD) file is created by the simulator. The VCD file keeps the switching activity of the signals of the netlist. Then, this VCD file is converted to a switching activity interchange format (SAIF) file, which is used by the Synopsys Power Compiler. As the final step, Power Compiler generates the power estimation reports by using the netlist information and SAIF file.

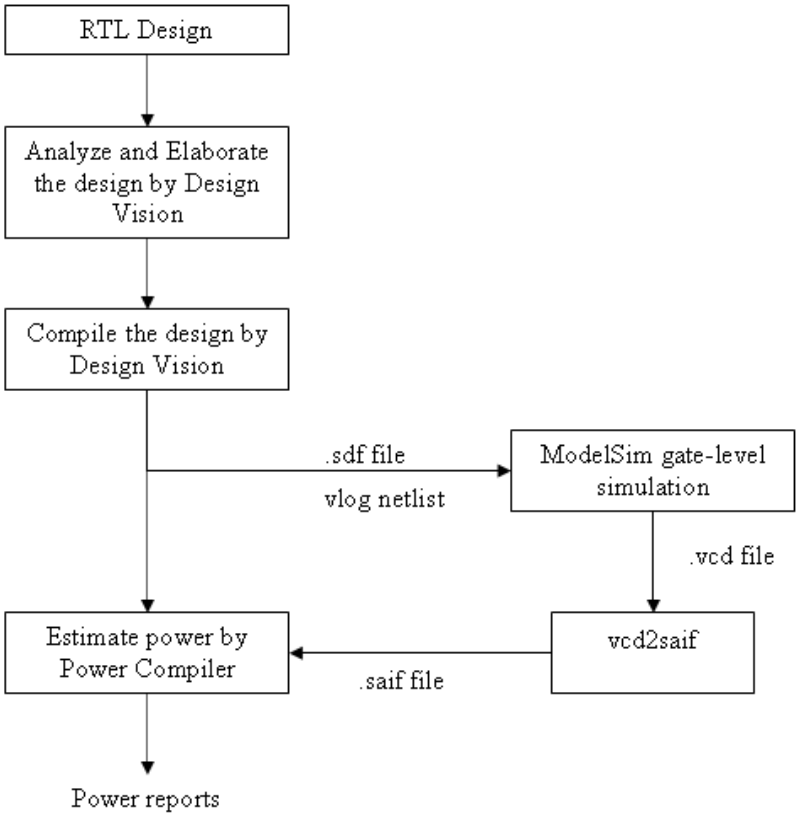


Figure 5.1: Power estimation flow

5.2 Results

Three different encryption-only NTRU designs were synthesized in order to find the best solution. In the first design, no enhancements have been applied to decrease the power consumption. In the second design, clock gating was applied to all possible registers and in the third design, partially rotating registers were used with clock gating. These designs will be denoted as *Enc1*, *Enc2* and *Enc3* respectively. Table 5.1 shows the area and power consumptions of these three designs. Equivalent gate counts of the designs were found by dividing the total area of the circuit by the area of one 2-input NAND gate. All of the area values represent the gate equivalents.

It can be seen from the results that clock gating regular registers does not decrease the power consumption of NTRU encryption. On the contrary the dynamic power consumption in *Enc2* has increased to 5.57 μW from 4.51 μW. This situation is caused by the continuous switching of the registers during encryption and the extra switching activity added by the gating circuits. On the other hand,

Table 5.1: Power and area consumption of encryption-only NTRU

Designs	Area			Power		
	Comb	Non-comb	Total	$P_{dyn}(\mu W)$	$P_{sta}(nW)$	$P_{tot}(\mu W)$
Enc1	680	2,537	3,217	4.51	12.6	4.52
Enc2	666	2,078	2,744	5.57	12.3	5.58
Enc3	776	2,107	2,884	1.72	13	1.74

results show that using clock gating with partially rotating registers reduces the dynamic power consumption dramatically. Moreover, considerably low static power consumptions were estimated as a result of the technology library which has *pW* as the leakage power unit.

Table 5.2 compares the designs *Enc1* and *Enc3* in detail. The rotating register in the design *Enc1* is the major source of area consumption (73.6 %) and power consumption (82.6 %). It can be seen from the results that using PRR with clock gating, reduces the power consumption of the design by 60 %. It also decreases the area consumption as a result of clock gating. Furthermore *Enc3* design consumes only 0.18 μW in the idle state. The best solution, *Enc3* consumes an area of

Table 5.2: Comparisons of the designs *Enc1* and *Enc3*

Designs	Modules	Area				Power			
		Comb	Non-comb	Total	(%)	$P_{dyn}(\mu W)$	$P_{sta}(nW)$	$P_{tot}(\mu W)$	(%)
Enc1	Controller	131	111	243	7.6	0.2	1.05	0.237	5.2
	Lut h	416	0	416	12.9	0.28	1.52	0.239	5.3
	PM	98	87	186	5.8	0.3	0.83	0.306	6.8
	$N \times 2$ Reg	31	2,338	2,369	73.6	3.72	9.17	3.74	82.6
	NTRU Top	680	2,537	3,217	100	4.51	12.6	4.52	100
Enc3	Controller	145	145	290	10.1	0.31	1.3	0.315	18.1
	Lut h	416	0	416	14.4	0.22	1.52	0.22	12.9
	PM	100	81	181	6.3	0.25	0.84	0.26	14.8
	$N \times 2$ PRR	113	1,881	1,993	69.1	0.9	9.33	0.938	54
	NTRU Top	776	2,107	2,884	100	1.72	13	1.74	100

2,884 gate equivalents. In all of the designs encryption takes $N \times (N+1) + N$ clock cycles. N clock cycles for loading of the random polynomial and $(N+1)$ clock cycles for the generation of one cipher text coefficient. For the implemented case, $N = 167$ and $f = 500$ kHz, the latency is 28,223 clock cycles which is equal to 56.44 ms.

In order to find the best solution for the encryption-decryption NTRU, the same route has been followed as the encryption-only case. Firstly, a plain version was synthesized, secondly clock gating was used for the all possible registers, and as the third case, PRR with clock gated registers was used. As expected, the

best solution has emerged as the last case. The most optimized design has an equivalent gate count of 10.5 kgates which was 12.3 kgates for the unoptimized design. Again the major source of the area consumption is the registers which occupy 84 % of the total area (See Table 5.3).

Table 5.3: Area consumption of encryption-decryption NTRU in GEs

Block	Comb	Non-comb	Total	(%)
Controller	231	172	403	3.8
Luts	717	0	717	6.8
PM	109	81	190	1.8
Mod-3 Unit	66	102	168	1.2
$N \times 2$	33	1877	1910	18.2
$N \times 7$	388	6473	6961	66.3
Others	104	42	146	1.4
Total	1651	8848	10500	100

Power consumption of the encryption-decryption NTRU was estimated for three different working states: Encryption, decryption and idle state. Table 5.4 shows these estimations as a comparison between the plain design and the optimized design. The major sources of the power dissipation in NTRU plain are the

Table 5.4: Power consumption of encryption-decryption NTRU

Designs	States	Power		
		$P_{dyn}(\mu W)$	$P_{sta}(nW)$	$P_{tot}(\mu W)$
NTRU Plain	Encryption	12.3	49.4	12.4
	Decryption	15.9	50.5	16
	Idle	10.1	49.7	10.2
NTRU Opt.	Encryption	5.93	46.6	5.98
	Decryption	6.04	50.8	6.11
	Idle	0.45	46.3	0.5

registers $N \times 2$ and $N \times 7$, which consume an average 90 % of the total power in all three states. But, it can be seen that in the optimized NTRU, using a $N \times 7$ PRR and a $N \times 2$ regular register with clock gating, reduces the power consumption more than 50 %. Moreover, because of the technology library, low static power consumptions were estimated also in these designs.

For the encryption-decryption NTRU design, encryption takes $N \times ((N+1)+1)+N$ clock cycles while decryption takes $2 \times N \times (N+(n+3)+1)+N$ clock cycles. The extra 1 clock cycle for encryption is needed to load the result into the result

register. In decryption, $(n+3)$ is the latency of reduction by 3 where n is the bit-length of the input value. And in both cases N clock cycles are needed for the *loading* state. For the implemented case, $N = 167$ and $f = 500$ kHz, encryption takes 56.78 ms (28,390 clock cycles) and decryption takes 119.23 ms (59,619 clock cycles).

5.3 Comparison

There is not much work in the literature focused on low power NTRU implementations. In the master's thesis of O'Rourke [29], a hardware core is designed which performs only NTRU polynomial multiplication. The design has a minimum gate count of 1483 in the case of using only one processing unit. But the caches to store the polynomials are not included in this gate count. So, our design which is capable of performing the whole NTRU encryption is superior to [29] when we consider only the gate counts. Unfortunately, no power consumption data is available for comparison. The latency estimations have been done with different parameter sets of NTRU and in different frequencies which prevents us to make fair comparisons. Another NTRU implementation was presented by Bailey *et al.* [16], and it performs only NTRU encryption. The design takes approximately 60,000 gates in a Xilinx Virtex 1000EFG860 FPGA which is outperformed by our design.

The most detailed low power NTRU implementation belongs to Kaps [5] but it only performs encryption. Table 5.5 compares our *Enc3* implementation and [5]. These two designs were synthesized at 500 kHz frequency and for the same NTRU parameter set, $(N,q,p) = (167,128,3)$. Our design and [5] have almost the same

Table 5.5: Comparison of *Enc3* and [5]

Design	Technology	Area			Power			Latency
		Comb.	Non-comb	Total	$P_{dyn}(\mu W)$	$P_{sta}(nW)$	$P_{tot}(\mu W)$	Clock cycles
[5]	$0.13\mu m$	523	2,327	2,850	4.03	15.1e+03	19.13	29,225
Our	$0.13\mu m$	776	2,107	2,884	1.72	13	1.74	28,223

gate counts. However our design outperforms [5] in dynamic power consumption and latency. Dynamic power consumption of [5] is more than 2 times bigger than our design. Since, [5] has not used a low leakage technology library we can not compare the static power consumptions. Both designs work at 500 kHz, our

design finishes encryption in 56.44 ms while in [5] encryption takes 58.45 ms. Thus, our design is 3.5 % faster than [5].

Apart from NTRU implementations, there also exists, a compact ECC implementation for low power applications which was presented by Batina *et al.* [32]. In this paper the best solution has a gate count of 6,718 (data memory is not included). However there are not any power and latency estimations for that solution, so another solution which has a gate count of 8,104 (data memory is not included) has been chosen for comparison. That solution including the memory elements, has 12 kgates which would be more appropriate to compare with our encryption-decryption NTRU design that has 10.5 kgates. In [32] latency is mentioned as 115 ms for $f = 500$ kHz, that is also close to the decryption latency of our design which is 119.23 ms for $f = 500$ kHz. Power consumption is estimated as lower than 30 μW in [32] while our solution has an estimation of 6 μW for both encryption and decryption.

6. SIDE CHANNEL ANALYSIS

Cryptographic algorithms which are ensured to be secure mathematically can still leak information about the secret data due to the defects of their implementations. Thus, it can still be possible to obtain the secret key of a cryptographic system which is implemented straightforwardly. Side channel analysis takes the advantage of implementation specific characteristics to infer the secret data. Side channel information can be the execution time of the system, the electromagnetic emission or power consumption of the circuit.

6.1 Power Analysis

The most popular side channel analysis for embedded systems is the power analysis, which uses the fact that power consumption of a circuit depends on the data which is processed. It was first proposed by Kocher *et al.* [33] in 1999. In [33] two methods are introduced which are called Simple Power Analysis (SPA) and Differential Power Analysis (DPA). In DPA only one bit is used to infer the secret information. After Kocher *et al.*, Messerges *et al.* [34,35] proposed a method called multiple-bit DPA which uses more than one bit to attack a circuit. In 2004, a more powerful method which is called Correlation Power Analysis (CPA) was suggested by Brier *et al.* [36].

6.1.1 Simple power analysis

Simple Power Analysis is a method that needs direct visual interpretation of the power measurements which are taken during the execution of a circuit. SPA can reveal the sequence of the instructions as well as the secret key [33, 37]. The vulnerability of the circuits against SPA comes from the easily distinguishable difference of their power consumption for different operations. For example, an operation which is performed according to the value of a secret key bit can reveal information about that secret key bit [37].

6.1.2 Differential power analysis

Differential Power Analysis attacks use several power traces (see Figure 6.1) which are measured while the circuit is active. Then these power traces are processed in order to reveal the secret information. In CMOS circuits $0 \rightarrow 1$ transitions consume more power than $1 \rightarrow 0$ transitions [38]. This difference helps an adversary to find the key.

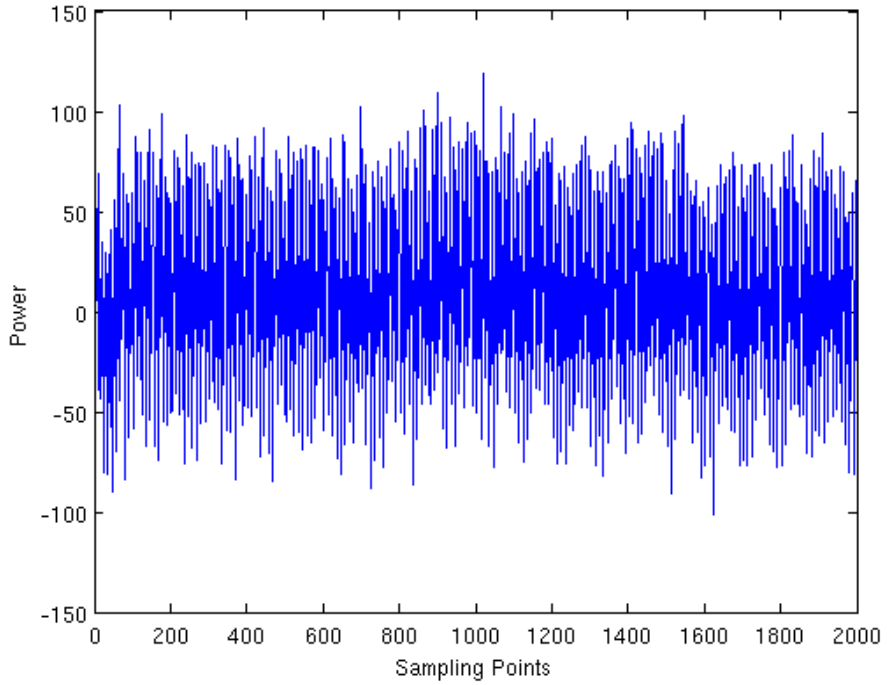
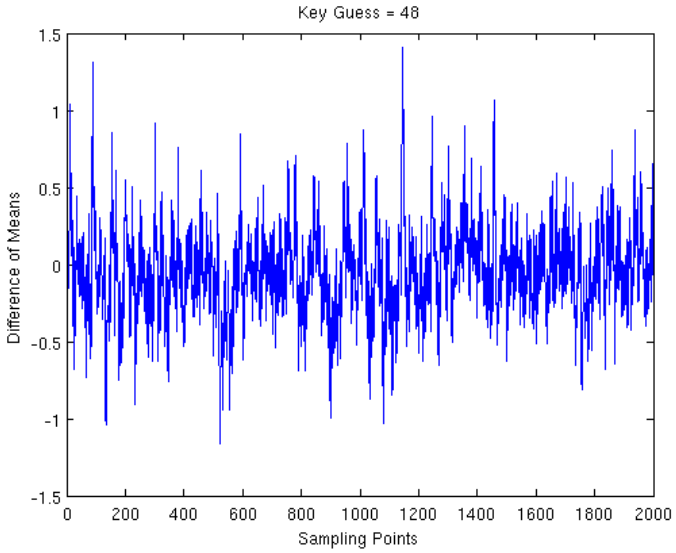
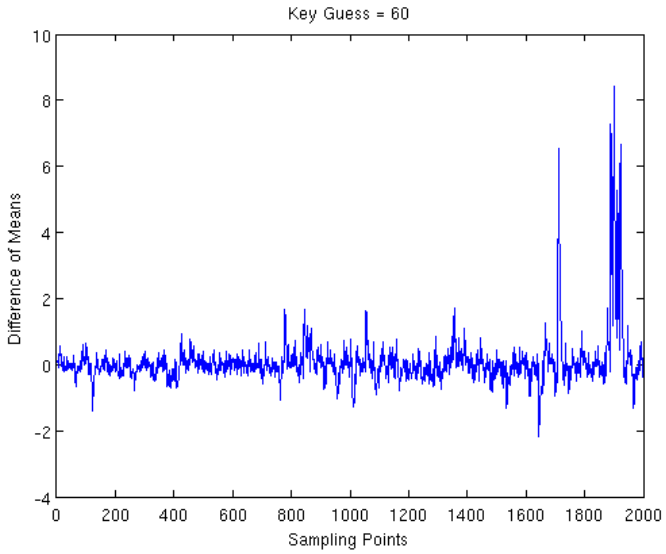


Figure 6.1: One power trace for an AES smartcard implementation

For example, if an adversary wants to attack an AES [10] implementation on a smartcard, he must first choose a one bit selection function D which is conveniently one bit of the output of a targeted sbox. Then he sends several plain texts and measures the power consumption of the circuit. After this step, he is ready to attack the sbox output of the implementation in the first AES round. He computes the value of the selection function D for the plain texts and for each key guess. If $D = 1$ he puts the corresponding measurement to the set $S1$, if $D = 0$ he puts the corresponding measurement to the set $S0$. Afterwards he computes the means (pointwise) of these sets, $M1$ and $M0$. Finally, he computes the difference of these means. For the correct key guess, a peak must appear in the difference of means as seen in the Figure 6.2.



(a)

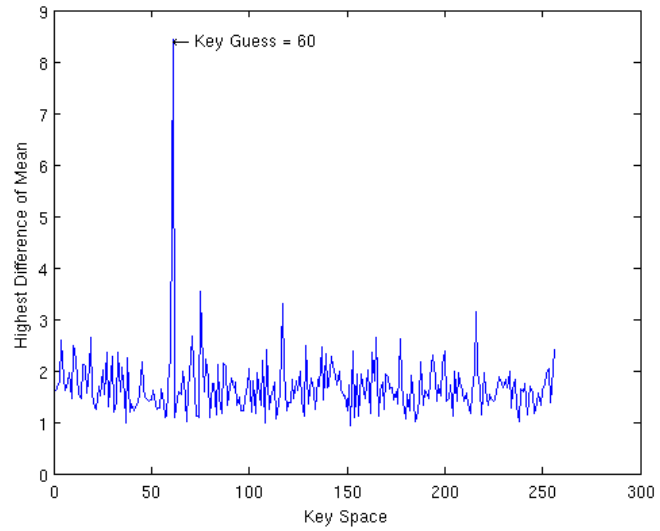


(b)

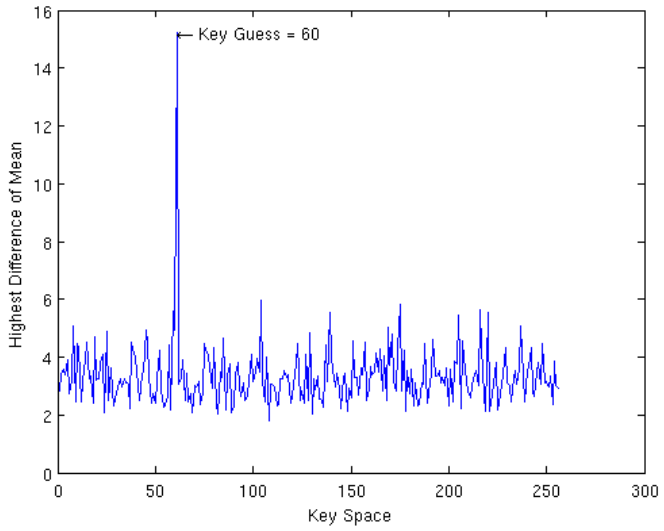
Figure 6.2: Difference of means for (a) a wrong key guess and for (b) the correct key guess

In [33] only one bit is used for the attack and the others are ignored. In order to improve the attack, it is proposed [34, 35] to use more than one bit to conduct the attack. This attack is called multiple-bit DPA. With this method, the signal-to-noise ratio of the differential traces can be increased and higher and more distinguishable peaks can be obtained than in the classical DPA. To attack the previous AES implementation, an adversary first chooses a $d-bit$ selection function which is d bits of the output of an sbox. Then he computes the values for the selection function as in the classical DPA. The adversary partitions the

power measurements into three sets. He puts the power measurements in the set $S1$ if all bits of d are one, in the set $S0$ if all bits of d are zero, else in the set $S2$. The values in the set $S2$ are not used. Finally he calculates the difference of the means of the sets $S1$ and $S0$. Figure 6.3 shows the results for both methods.



(a) Classical DPA



(b) Multiple-bit DPA

Figure 6.3: DPA results for the AES implementation

Another method, which is more effective than the former methods, is called Correlation Power Analysis (CPA) [36]. It is based on finding the correlation coefficient between power measurements and the power consumption model of the system. The number of bits switching from one state to another highly affects the power consumption in the circuit [37]. The power dissipation of the circuit

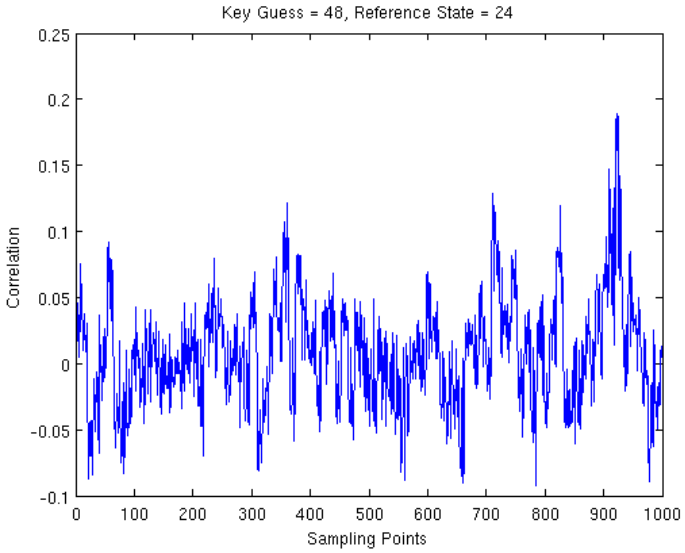
can be modeled with the Hamming Distance Model. The Hamming distance is the number of flipping bits from a reference state R to another state D. Thus, the number of the switching bits equals $H(R \oplus D)$ where H represents the Hamming weight function which is simply the number of bits that are set to '1'. Using the Hamming Distance Model and the plain text values, the power consumption of the circuit is modeled for all the possible reference states and the key guesses. Then, the correlation coefficients between the power measurements and the predicted power consumption values are calculated. The Pearson correlation coefficient between the power measurements and the predictions can be estimated by the following equation [36]:

$$\hat{\rho}_{WH}(R) = \frac{N\sum W_i H_{i,R} - \sum W_i \sum H_{i,R}}{\sqrt{N\sum W_i^2 - (\sum W_i)^2}\sqrt{N\sum H_{i,R}^2 - (\sum H_{i,R})^2}} \tag{6.1}$$

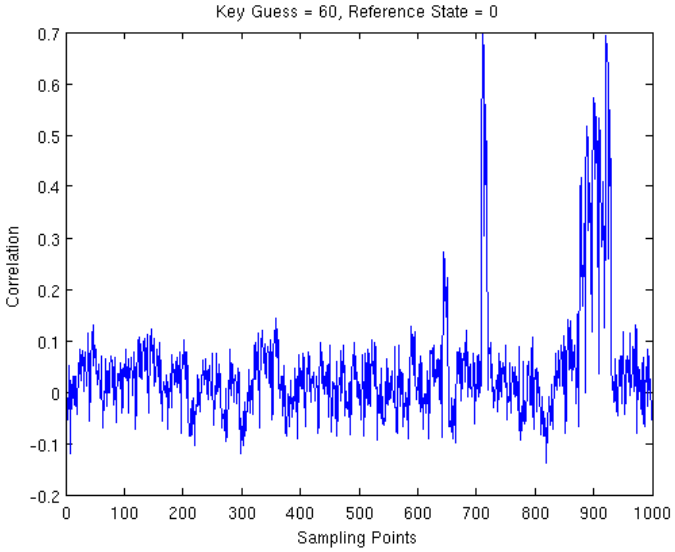
Here, H denotes the Hamming Distance Model of the circuit for the reference state R. W represents the power traces for N measurements. The summations are taken over N samples and the correlation coefficient has to be estimated for each time slice within the traces W_i separately. The highest correlation coefficient appears for the correct key and reference state guess. The result of the CPA for the AES smartcard implementation is shown in Figure 6.4.

6.2 Power Analysis Attacks on NTRU

In order to test the security of the NTRU implementation, CPA attacks were conducted on the encryption-decryption NTRU circuit. The NTRU design which is used for DPA attacks, has only operand isolation as the power reduction method. The first multiplication of decryption which is $a = f * e \pmod q$ $(e = e_o + e_1X + \ldots + e_{N-1}X^{N-1},\ f = f_0 + f_1X + \ldots + f_{N-1}X^{N-1})$ is chosen for the attack. The coefficients of $a(X)$ are computed serially in decreasing order, starting from a_{N-1}. Each coefficient is computed with a multiply and accumulate approach. In each clock cycle the architecture computes the product of two coefficients, adds it to the previously computed value which is fed back from a register, and updates the register with the new value. The register which stores the intermediate values is set to all zeros before starting to compute a new coefficient of a. For example, the computation of a_{N-1} starts with the



(a) Wrong key and reference state



(b) Correct key and reference state

Figure 6.4: CPA results for the AES implementation

computation of the partial product $e_0 f_{N-1}$. Then, the first partial product is loaded to the register and the second partial product $e_1 f_{N-2}$ is computed. In the following clock cycle the sum of the value of the register which is $e_0 f_{N-1}$ and the second partial product is loaded to the register and the third partial product is computed. The computation proceeds similarly till reaching the result a_{N-1}.

6.2.1 Measurement setup

NTRU design was implemented on a Xilinx Virtex-II (xc2vp7-5fg456) FPGA. The design occupies 1,167 slices which is 23 % of the total slice number. The FPGA

was supplied with an external DC power supply (3.3V). The clock frequency of the circuit was 100 kHz. The current was measured with a current probe which is placed serially between the power supply and the V_{DD} input of the FPGA. The output of the probe was sampled with an oscilloscope with 1 GHz bandwidth at a rate of 40 MS/s. Then, the sampled data was transferred to a PC for analysis.

In the power analysis, $N = 16,384$ power traces were used. Figure 6.5, shows one of these power traces. The power trace contains six clock cycles which covers an idle cycle and the first five cycles of the first multiplication of decryption.

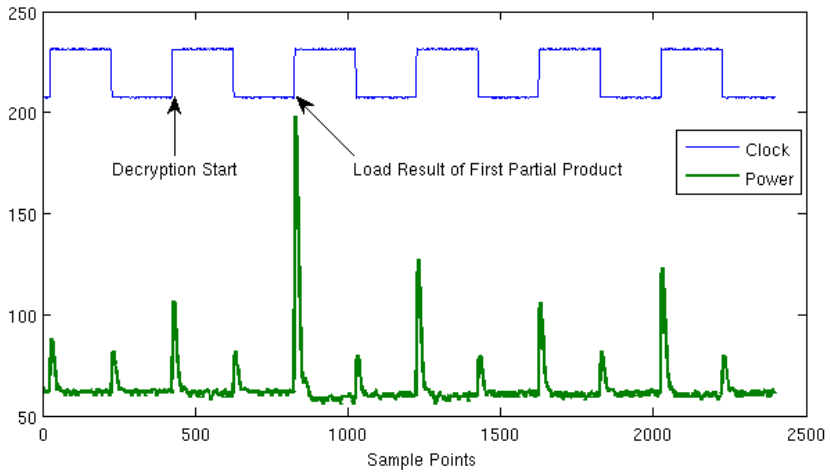


Figure 6.5: One power trace for the NTRU implementation

The cipher text values were given by another circuit which is implemented in the same FPGA. In order to obtain uniformly distributed inputs, e_{0_i} and e_{1_i} were chosen as $\lfloor i/128 \rfloor$ (mod 128) and i (mod 128) respectively where i is the measurement number. All other coefficients of e were set to all zeros in order to reduce algorithmic noise.

6.2.2 Attack and results

The attack aims to recover the coefficients of the private key f one by one. In order to conduct the attack, the power consumption of the register which is used during the multiplication and accumulation of the partial products, is analysed. The dissipated power is expected to be correlated with the Hamming distance of the register between its *new* and *previous* values. Thus, the power consumption predictions H_i of the circuit are calculated with the function $H_i = H(new_i \oplus previous_i)$ for the i^{th} cipher text. Here, a problem occurs if the relevant

coefficient of f is zero. The situation $f_i = 0$ results in no change in the value of the register which makes the Hamming distance zero. Since the register value does not vary in this case, DPA can not be applied successfully. Hence, for the power model only key guesses -1 and 1 are tested. If none of them leads to a high correlation coefficient, then that key coefficient is assumed to be 0. This hypothesis can be confirmed or rejected when attacking the next coefficient, since the new value of the register depends on the all previous values which involve the key coefficients.

The coefficient f_{N-1} is used in the first partial product which is $e_{0_i} f_{N-1}$. Thus, to recover that coefficient, the clock cycle which the result of the product is loaded to the register, must be analysed. Since, for a new computation, the register starts with the value zero, the Hamming distance of the register becomes $H_i = H(e_{0_i} f_{N-1} \oplus 0) = H(e_{0_i} f_{N-1})$. Similarly for the second coefficient f_{N-2}, the Hamming distance is computed with the equation $H_i = H(e_{0_i} f_{N-1} + e_{1_i} f_{N-2} \oplus e_{0_i} f_{N-1})$.

Figure 6.6 shows the attack results for f_{N-1} and Figure 6.7 shows the results for f_{N-2}. In the figures it is clearly seen that the correct hypothesis $f_{N-1} = -1$, $f_{N-2} = 1$ leads to the highest correlation coefficient. The result is also confirmed by finding the correlation coefficient for the pair (f_{N-1}, f_{N-2}) together, instead of conducting the attack coefficient by coefficient. The result is shown in Figure 6.8. The remaining coefficients can be obtained with the same method.

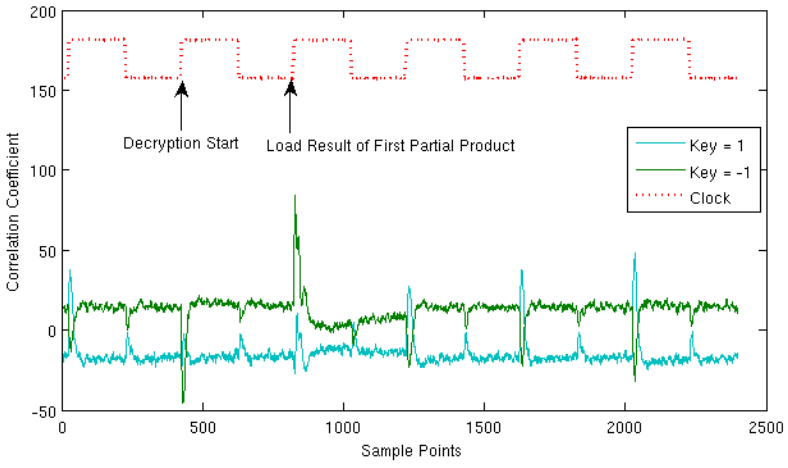


Figure 6.6: Correlation traces for f_{N-1}

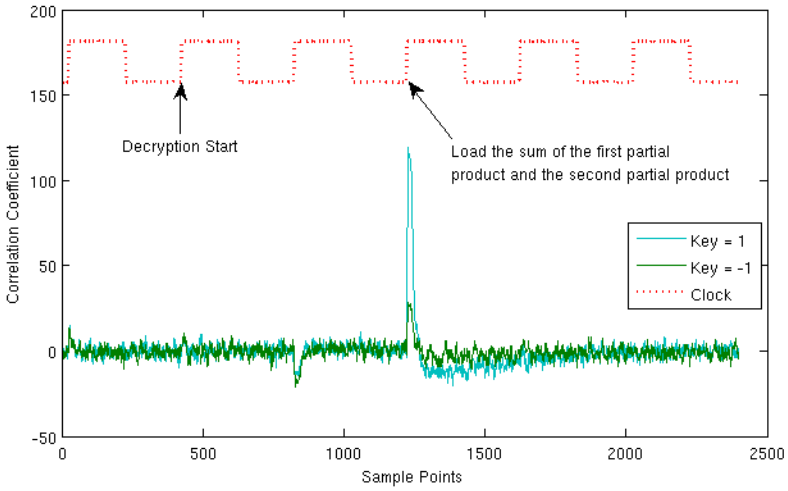


Figure 6.7: Correlation traces for f_{N-2}

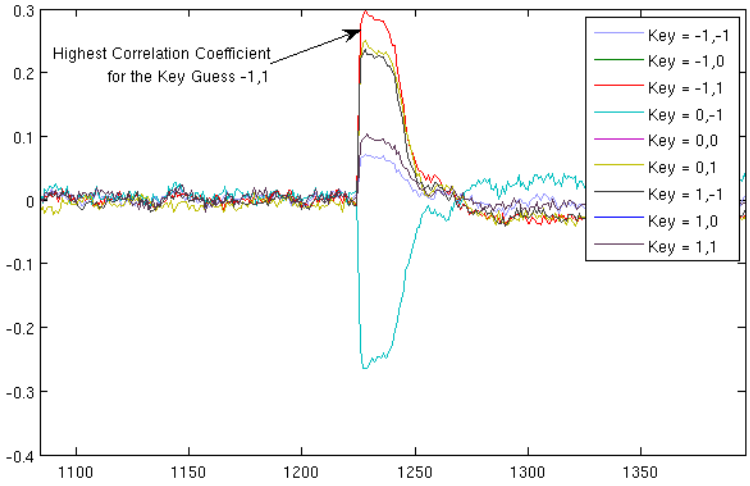


Figure 6.8: Correlation traces for the pair (f_{N-1}, f_{N-2})

6.3 DPA Countermeasures for Public-Key Cryptosystems

DPA attacks give the opportunity of inferring the secret keys easily in contrast with the classical cryptoanalysis methods. In order to prevent DPA, many techniques have been proposed by the researchers. The main aim of these methods is to randomize the input data or the sequence of the calculations to diminish the correlation between the power consumption of the circuit and the known data which is assumed to be open to the adversary.

RSA [13] is one of the most popular public-key cryptosystems but carelessly done implementations of RSA is not immune to DPA. In RSA, encryption and decryption are performed by modular exponentiation. If m is assumed as the plaintext message and e as the public-key, then the encryption of RSA is performed as follows:

$$c = m^e \text{ (mod n)} \tag{6.2}$$

Decryption is performed with the modular exponentiation of ciphertext c with the secret-key d:

$$m = c^d \text{ (mod n)} \tag{6.3}$$

The parameter $n = pq$ where p and q are prime numbers. The keys satisfy the equation $d = e^{-1} (\text{mod}\phi(n))$ where $\phi(n) = (p-1)(q-1)$ is the Euler totient function.

RSA implementations can be made DPA resistant in two ways. The first method is two randomize the input data and/or the secret exponent d [39]. The classical method is to use $\hat{d} = d + r\phi(n)$ during modular exponentiation where r is a random number. Calculation of $c^{\hat{d}}$ (mod n) gives the correct plaintext since $x^{\phi(n)} = 1$ (mod n). With this method, the power trace of the circuit during a decryption is not correlated with d anymore, but correlated with $\hat{d}$ which is updated for every decryption.

The other method is to randomize the algorithm used for calculation of the exponentiation. Walter proposed an algorithm called MIST to randomize the exponentiation which is a modified version of the binary square and multiply algorithm [40]. The MIST algorithm generates randomly different addition chains to perform a specific modular exponentiation. Furthermore, this algorithm

disables the attacks which are based on recognising the repeated use of the same pre-computed multipliers during an individual exponentiation. Another exponent randomization algorithm is presented by Chevallier-Mames [41] which mentions the concept of self-randomized exponentiation. In the algorithm the exponent d is used itself as the source of randomness. During the exponentiation the exponent d is randomly splitted to smaller parts. The exponentiation is performed with these splitted parts instead of the secret exponent d. The method is not only applicable for RSA groups but also applicable for elliptic curve groups.

Elliptic Curve Cryptography (ECC) [14] is another commonly used public-key cryptosytem. However the straightforward implementations of ECC are also not resistant to DPA attacks. An elliptic curve is the set of points (x,y) that are the solutions of the equation below which is defined over a field K:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{6.4}$$

Elliptic Curve Cryptography is based on the scalar multiplication of an elliptic curve point P with a scalar d. The scalar multiplication dP can be computed simply by adding the point P with itself d times:

$$dP = \underbrace{P+P+P+\cdots+P}_{\text{d-times}} \tag{6.5}$$

Addition formulas of different curve points (P_1+P_2) and the same curve points (P_1+P_1) differs from each other. Detailed information about the elliptic curves can be found in [42].

The DPA attacks on elliptic curves are focused on finding the scalar d which is the secret key of the system. In order to prevent these attacks three methods are proposed in [43]. The first method is to randomize the scalar d. It is proposed to multiply the point P with $d' = d + k\#\varepsilon$ instead of d. $\#\varepsilon$ represents the number of points on the curve and k is a scalar. The multiplication satisfies the equality $Q = d'P = dP$ since $\#\varepsilon P = \mathcal{O}$ where $\mathcal{O}$ is the identity element of the group. The second method is to blind the point P. It is done by adding a random point R to P for which we know $S = dR$. Scalar multiplication is performed by computing $d(R+P)$ and then subtracting $S = dR$ to obtain $Q = dP$.

It is also possible to calculate $Q = dP$ by using the projective coordinates [42] of the point P. The projective coordinates $P(X,Y,Z)$ of a point $P(x,y)$ can be given as:

$$x = \frac{X}{Z} \quad y = \frac{Y}{Z} \tag{6.6}$$

The last method is to use randomized projective coordinates. The projective coordinates of a point is not unique since $(X,Y,Z) = (\lambda X, \lambda Y, \lambda Z)$ for $\lambda \neq 0$. Thus, by using a different, randomly chosen λ for each scalar multiplication, randomization of the point P can be achieved.

Another approach in strengthening the ECC implementations is to randomize the scalar multiplication algorithm. Oswald *et. al.* proposed a countermeasure in [44] which uses randomized addition-subtraction chains in the scalar multiplication algorithm. They present two algorithms which uses different addition-subtraction chains to calculate the scalar multiplication. Then, they randomize their algorithms by inserting a random decision. A random variable is chosen during the computation and according to the value of that variable the classical binary algorithm or one of their proposed algorithms is used to proceed the computation.

6.4 Power Analysis of DPA Resistant NTRU

In this thesis a DPA resistant NTRU implementation which is intellectual property of ESAT/COSIC Katholieke Universiteit Leuven, is also analyzed. The implementation under test was using input randomization to resist DPA. The circuit under test was implemented on a Xilinx VirtexE (xcv1000e-7hq240) FPGA. The clock frequency of the circuit was 2 MHz. 10000 points were sampled with a current probe for each power trace. Sampling is done with an oscilloscope with 1 GHz bandwidth at a 2 GS/s sampling rate.

The methodology in Section 6.2.2 is also applied here. In the power analysis $N = 16,384$ traces were used. One trace contains nine clock cycles. At every $(k \times 1000)^{th}$ point a rising edge occurs. Fig. 6.9 shows one power trace for the protected NTRU implementation.

For this attack correlations are calculated differently. Firstly, nine intervals which include nine rising edges of the clock was chosen. Then the difference of the

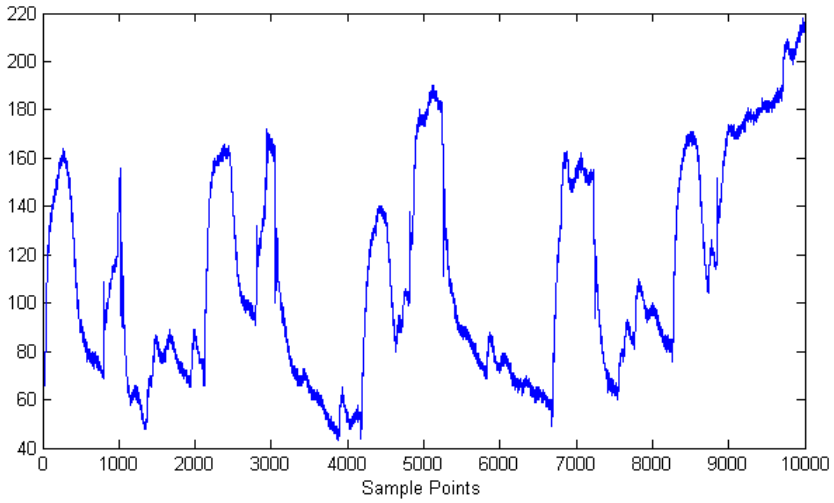


Figure 6.9: One power trace for the protected NTRU implementation

maximum and the minimum value in the interval is calculated to find the current change during the rising edge of the clock. Hence, nine points are obtained for each trace. First point belongs to the start point of the decryption, second point belongs to the loading of the partial sum $e_{0_i} f_{N-1}$ and the third point belongs to the loading of the partial sum $e_{0_i} f_{N-1} + e_{1_i} f_{N-2}$. Finally, correlation coefficients are computed between these points and the hamming distance model.

The computed correlation coefficients are plotted versus the number of traces. Figure 6.10 shows the result for the coefficient f_{N-1}. It is seen that the key guess $f_{N-1} = -1$ shows a higher correlation curve after 8200 traces. The result for the second coefficient f_{N-2} is shown in Figure 6.11. It is obvious that only after 100 traces, the key guess $f_{N-2} = 1$ has the higher correlation coefficient.

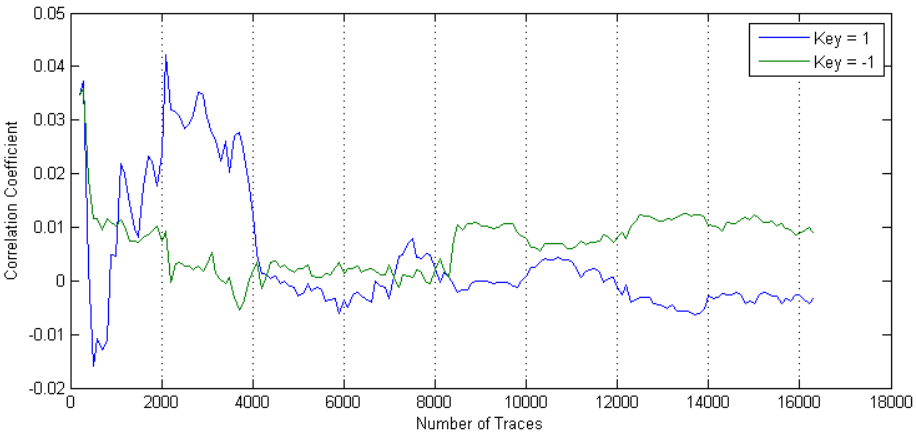


Figure 6.10: Correlation coefficients for f_{N-1}

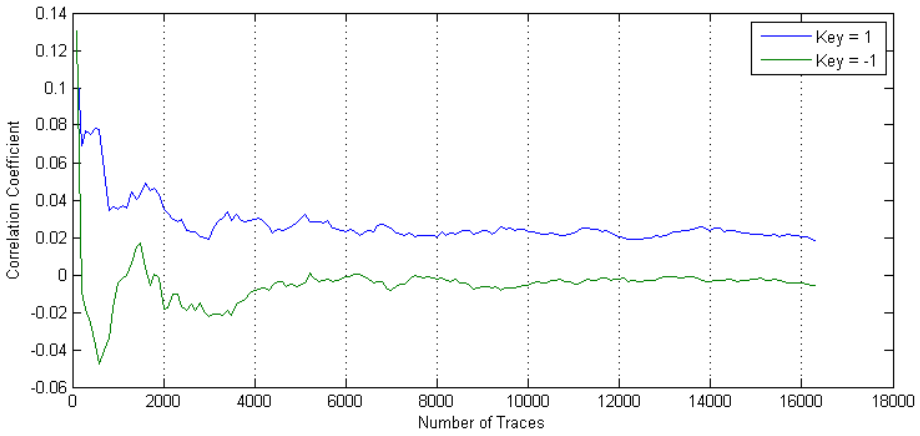


Figure 6.11: Correlation coefficients for f_{N-2}

7. CONCLUSION

In this dissertation, compact and low power NTRU implementations that are suitable for pervasive security applications such as RFID and WSN systems are presented. One design is capable of only performing NTRU encryption and the other one can perform both encryption and decryption. Encryption-only NTRU design consumes only 2.8 kgates, which makes it a very compact solution for resource restricted devices. The encryption takes 56.44 ms at 500 kHz which is 3.5 % faster than the best previous work. As a result of the low leakage technology library, static power consumption is considerably low and the design consumes 1.72 μW of dynamic power during one encryption operation. This presents a saving of a factor more than 2, when compared with the previous work. This work is also the first one to present a complete NTRU design with encryption/decryption circuitry. Encryption-decryption NTRU has a gate count of 10.5 kgates. The optimized design has the following results, 5.93 μW, 6.04 μW and 0.45 μW for dynamic power consumption during encryption, decryption and idle state, respectively. The latency for encryption and decryption, is 56.78 ms and 119.23 ms at 500 kHz respectively. Moreover, the security of the implementation against side channel analysis attacks has been explored and it has been shown that the implementation is not immune to Differential Power Analysis (DPA) attacks. The secret key was revealed by using 16,384 power measurements, but we expect that less measurements suffice. A protected implementation of the NTRU design which is intellectual property of ESAT/COSIC Katholieke Universiteit Leuven, is also tested for side channel resistance. The keys of the protected implementation were inferred with the same attack too.

As the future work of this thesis, there are opportunities to improve both the performance and the security of the designs. It is possible to speed up the designs by adding more Polynomial Multiplier (PM) units with a small change at the rest of the circuit. This will enable the designs to compute more than one

ciphertext coefficients in parallel at a time which will reduce the encryption and decryption latency. It is also possible to perform decryption with one polynomial multiplication if the design parameters (N, q, p) are selected in another way which also brings the necessity of redesigning some modules. Finally, to improve the security of the designs, effective countermeasures against DPA attacks must be explored and implemented.

REFERENCES

[1] **Juels, A.**, 2006. RFID Security and Privacy: A Research Survey, *IEEE Journal on Selected Areas in Communications*, **24(2)**, 381–394.

[2] **Peris-Lopez, P.**, **Hernandez-Castro, J.C.**, **Estevez-Tapiador, J. and Ribagorda, A.**, 2006. RFID Systems: A Survey on Security Threats and Proposed Solutions, 11th IFIP International Conference on Personal Wireless Communications PWC'06, volume 4217 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 159–170.

[3] **Ayoade, J.**, 2007. Privacy and RFID Systems: Roadmap to solving security and privacy concerns in RFID systems, *Computer Law and Security Report*, **23(6)**, 555–561.

[4] **Culler, D.**, **Estrin, D. and Srivastava, M.**, 2004. Guest Editors' Introduction: Overview of Sensor Networks, *Computer*, **37(8)**, 41–49.

[5] **Kaps, J.**, 2006. Cryptography for Ultra-Low Power Devices, Ph.D. thesis, Worcester Polytechnic Institute.

[6] **Gaubatz, G.**, **Kaps, J.P.**, **Öztürk, E. and Sunar, B.**, 2005. State of the Art in Ultra-Low Power Public Key Cryptography for Wireless Sensor Networks, Third IEEE Int. Conf. Pervasive Comput. Commun. Workshops, volume v2005, IEEE Computer Society, pp. 146–150.

[7] **Hoffstein, J.**, **Pipher, J. and Silverman, J.H.**, 1998. NTRU: A Ring-Base Public Key Cryptosystem, **J.P. Buhler**, editor, Algorithmic Number Theory (ANTS III), Lecture Notes in Computer Science, volume 1423, Springer-Verlag, Berlin, pp. 267–288.

[8] **Stinson, D.R.**, 2006. Cryptography Theory and Practice, Chapman & Hall/CRC, Taylor & Francis Group, 3^{rd} edition.

[9] **Menezes, A.J.**, **van Oorschot, P.C. and Vanstone, S.A.**, 1996. Handbook of Applied Cryptography, CRC Press.

[10] **FIPS 197: Advanced Encryption Standard (AES)**, 2001, National Institute of Standards and Technology (NIST), http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[11] **FIPS 46-3: Data Encryption Standard (DES)**, reaffirmed 1999, National Institute of Standards and Technology (NIST), http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf.

[12] **Diffie, W. and Hellman, M.E.**, 1976. New Directions in Cryptography, *IEEE Transactions on Information Theory*, **IT-22(6)**, 644–654.

[13] **Rivest, R.L.**, **Shamir, A. and Adleman, L.**, 1978. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, *Communications of the ACM*, **21(2)**, 120–126.

[14] **Miller, V.**, 1985. Uses of Elliptic Curves in Cryptography, **H.C. Williams**, editor, Advances in Cryptology: Proceedings of CRYPTO'85, number 218 in Lecture Notes in Computer Science, Springer-Verlag, pp. 417–426.

[15] The NTRU Public Key Cryptosystem A-Tutorial, http://www.ntru.com/cryptolab/tutorials.htm.

[16] **Bailey, D.**, **Coffin, D.**, **Elbirt, A.**, **Silverman, J. and Woodbury, A.**, 2001. NTRU in Constrained Devices, Cryptographic Hardware and Embedded Systems, Paris, France.

[17] **Howgrave-Graham, N.**, **Silverman, J.H. and Whyte, W.**, 2005. Choosing Parameter Sets for NTRUEncrypt with NAEP and SVES3, Topics in Cryptology CT-RSA 2005, Lecture Notes in Computer Science, volume 3376, Springer, Berlin, pp. 118–135.

[18] **Silverman, J.H.**, 1998. Invertibility in Truncated Polynomial Rings, Technical report, NTRU Cryptosystems.

[19] **Silverman, J.H.**, 1999. Almost Inverses and Fast NTRU Key Creation, Technical report, NTRU Cryptosystems.

[20] **Pedram, M. and Rabaey, J.**, 2002. Power Aware Design Methodologies, Kluwer Academic Publishers, 1^{st} edition.

[21] **Rabaey, J. and Pedram, M.**, 1996. Low Power Design Methodologies, Kluwer Academic Publishers, 2^{nd} edition.

[22] **Keating, M.**, **Flynn, D.**, **Aitken, R.**, **Gibbons, A. and Shi, K.**, 2007. Low Power Design Methodology Manual For System-on-Chip Design, Springer, 1^{st} edition.

[23] Synopsys, Inc., 2006. Power Compiler User Guide.

[24] Gezel Tutorial, http://rijndael.ece.vt.edu/gezel2/index.php.

[25] **Omondi, A.R.**, 1994. Computer Arithmetic Systems Algorithms, Architecture and Implementation, Prentice Hall, 1^{st} edition.

[26] **Koren, I.**, 2002. Computer Arithmetic Algorithms, A K Peters, 2^{nd} edition.

[27] 2007, www.zenoli.net/category/mathematics/.

[28] Synopsys, Inc., 2006. Design Compiler Tutorial Using Design Vision.

[29] **O'Rourke, C.M.**, 2002. Efficient NTRU Implementations, Master's thesis, Worcester Polytechnic Institute.

[30] **Gaubatz, G.**, **Kaps, J.P. and Sunar, B.**, 2004. Public Key Cryptography in Sensor Networks—Revisited, **H. Hartenstein**, **C. Castellucia**, **C. Paar and D. Westhoff**, editors, 1st European Workshop on Security in Ad-Hoc and Sensor Networks (ESAS 2004), volume 3313 of *Lecture Notes in Computer Science (LNCS)*, Springer, Heidelberg, pp. 2–18.

[31] **Kaps, J.P.**, **Gaubatz, G. and Sunar, B.**, 2007. Cryptography on a Speck of Dust, *Computer*, **40(2)**, 38–44.

[32] **Batina, L.**, **Mentens, N.**, **Sakiyama, K.**, **Preneel, B. and Verbauwhede, I.**, 2006. Low-Cost Elliptic Curve Cryptography for Wireless Sensor Networks, 4th European Workshop on Security and Privacy in Ad hoc and Sensor Networks, Lecture Notes in Computer Science, volume 4537, Springer-Verlag, pp. 6–17.

[33] **Kocher, P.**, **Jaffe, J. and Jun, B.**, 1999. Differential Power Analysis, **M. Wiener**, editor, Advances in Cryptology: Proceedings of CRYPTO'99, number 1666 in Lecture Notes in Computer Science, Springer-Verlag, pp. 388–397.

[34] **Messerges, T.S.**, **Dabbish, E.A. and Sloan, R.H.** Investigations of Power Analysis Attacks on Smartcards, pp. 151–162, *citeseer.ist.psu.edu/messerges99investigations.html*.

[35] **Messerges, T.S.**, 2000. Power Analysis Attacks and Countermeasures for Cryptographic Algorithms, Ph.D. thesis, University of Illinois at Chicago.

[36] **Brier, E.**, **Clavier, C. and Olivier, F.**, 2004. Correlation Power Analysis with a Leakage Model, **M. Joye and J.J. Quisquater**, editors, Proceedings of 6th International Workshop on Cryptographic Hardware and Embedded Systems (CHES), number 3156 in Lecture Notes in Computer Science, Springer-Verlag, pp. 16–29.

[37] **Ors, S.B.**, **Oswald, E. and Preneel, B.**, 2003. Power-Analysis Attacks on an FPGA - First Experimental Results, Cryptographic Hardware and Embedded Systems - CHES 2003, volume 2799 of *Lecture Notes in Computer Science*, Springer Berlin/Heidelberg, pp. 35–50.

[38] **Örs Yalçın, S.B.**, 2005. Hardware Design of Elliptic Curve Cryptosystems and Side-Channel Attacks, Ph.D. thesis, Katholieke Universiteit Leuven.

[39] **Kocher, P.**, 1996. Timing attacks on implementations of Diffie-Hellman, RSA, DSS and other systems, **N. Koblitz**, editor, Advances in Cryptology: Proceedings of CRYPTO'96, number 1109 in Lecture Notes in Computer Science, Springer-Verlag, pp. 104–113.

[40] **Walter, C.D.**, 2002. MIST: An Efficient, Randomized Exponentiation Algorithm for Resisting Power Analysis, CT-RSA '02: Proceedings of the The Cryptographer's Track at the RSA Conference on Topics in Cryptology, volume 2271 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 53–66.

[41] **Chevallier-Mames, B.**, 2004. Self-Randomized Exponentiation Algorithms, Topics in Cryptology - CT-RSA 2004, volume 2964 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 236–249.

[42] **Hankerson, D.**, **Menezes, A. and Vanstone, S.**, 2004. Guide to Elliptic Curve Cryptography, Springer, 1^{st} edition.

[43] **Coron, J.S.**, 1999. Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems, CHES '99: Proceedings of the First International Workshop on Cryptographic Hardware and Embedded Systems, volume 1717 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 292–302.

[44] **Oswald, E. and Aigner, M.**, 2001. Randomized Addition-Subtraction Chains as a Countermeasure against Power Attacks, CHES '01: Proceedings of the Third International Workshop on Cryptographic Hardware and Embedded Systems, volume 2162 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 39–50.

CURRICULUM VITAE

Name Surname: Ali Can ATICI

Place and Date of Birth: Adana / 24-11-1983

Universities and
Colleges attented: Istanbul Technical University

Publications:

- **A. C. Atici**, L. Batina, J. Fan, I. Verbauwhede and S. B. Ors Yalcin, "Low-cost Implementations of NTRU for pervasive security", *In Proceedings of 19th IEEE Conference on Application-specific Systems, Architectures and Processors (ASAP)*, Belgium, 2008

- **A. C. Atici**, L. Batina, B. Gierlichs, and I. Verbauwhede, "Power analysis on NTRU implementations for RFIDs: First results", *In Workshop on RFID Security 2008*, 11 pages, 2008