Open access • Proceedings Article • DOI:10.1109/ASAP.2008.4580158

Low-cost implementations of NTRU for pervasive security — Source link

A.C. Atici, Lejla Batina, Junfeng Fan, Ingrid Verbauwhede ...+1 more authors

Institutions: Katholieke Universiteit Leuven, Istanbul Technical University

Published on: 02 Jul 2008 - Application-Specific Systems, Architectures, and Processors

Topics: NTRU, Encryption, Clock gating, Cryptosystem and Cryptography

Low-cost Implementations of NTRU for pervasive security*

Ali Can Atıcı
Istanbul Technical University
Institute of Science and Technology
aticial@itu.edu.tr

Lejla Batina
Katholike Universiteit Leuven
ESAT/COSIC and IBBT
lejla.batina@esat.kuleuven.be

Junfeng Fan
Katholike Universiteit Leuven
ESAT/COSIC and IBBT
junfeng.fan@esat.kuleuven.be

Ingrid Verbauwhede
Katholike Universiteit Leuven
ESAT/COSIC and IBBT
ingrid.verbauwhede@esat.kuleuven.be

S. Berna Örs Yalçın
Istanbul Technical University
Faculty of Electrical and Electronics Engineering
siddika.ors@itu.edu.tr

Abstract

NTRU is a public-key cryptosystem based on the shortest vector problem in a lattice which is an alternative to RSA and ECC. This work presents a compact and low power NTRU design that is suitable for pervasive security applications such as RFIDs and sensor nodes. We have designed two architectures, one is only capable of encryption and the other one performs both encryption and decryption. The strategy for the designs includes clock gating of registers, operand isolation and precomputation. This work is also the first one to present a complete NTRU design with encryption/decryption circuitry. Our encryption-only NTRU design has a gate-count of 2.8 *kgates and dynamic power consumption of* $1.72\,\mu W$*. Moreover, encryption-decryption NTRU design consumes about* $6\,\mu W$ *dynamic power and consists of* 10.5 *kgates.*

1. Introduction

The feasibility of Public-key (PK) solutions for RFIDs and sensor networks is an open research problem due to severe limitations in costs, area and power. RFID tags and sensor nodes are extreme examples as they imply very low budget for the number of gates, power, bandwidth *etc.* whilst they sometimes require security solutions. Implementations of Public-key Cryptography (PKC) are very difficult in those environments as PKC deploys computationally demanding operations. 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 [3].

In this paper, we present low-cost implementations of the Public-key algorithm NTRU, which are viable for RFIDs and sensor nodes. Section 2 investigates the previous implementations of low-cost PKC. Section 3 summarizes the basics of NTRU algorithm such as key generation, encryption and decryption. Section 4 gives brief information about the low-power design methods used in the designs. Section 5 gives the details of the architecture of encryption-only NTRU whereas, section 6 shows the architecture of encryption/decryption NTRU. Section 7 discusses the power, area and latency results of our implementation and previous implementations. And finally, Section 8 concludes the paper.

2. Previous work

To the best of our knowledge very few papers discuss the possibility for PKC (specifically NTRU) for RFIDs and sensor nodes. In the master's thesis of O'Rourke [7], a hardware core is designed which performs only NTRU polynomial multiplication. It has a gate count of minimum 1483

*This work is funded partially by IBBT, Katholieke Universiteit Leuven (OT/06/40) and FWO projects (G.0300.07 and G.0450.04). This work was supported in part by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy), by the EU IST FP6 projects (ECRYPT) and by the IBBT-QoE project of the IBBT.

gates, but the design is not optimized for low-cost and there is no power consumption data. Another NTRU design is implemented by Bailey *et al*. [1], and it performs only encryption. The design has approximately 60 000 gates in a Xilinx Virtex 1000EFG860 FPGA. The most detailed low-cost implementation of NTRU is realized by Kaps [6]. The author investigated implementations of two algorithms: Rabin's scheme and NTRUEncrypt. It is concluded that NTRUEncrypt features a suitable low-power and small footprint solution with a total complexity of 3000 gates and power consumption of less than 20 μW at 500 kHz. As one would expect, he showed that Rabin's scheme is not a feasible solution. Apart from NTRU implementations, there also exist a compact ECC implementation in [2]. In this work, the best solution has 6718 gates for modular arithmetic unit and control unit (data memory not included), and it consumes power less than 30 μW at 500 kHz.

3. NTRU algorithm

NTRU [4] 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 is carried out in a normal way by adding the coefficients that have the same degree while multiplication is carried out in the following way. 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 [1] and denoted with the $*$ symbol. Thus, the product of two polynomials a and b

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

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

can be calculated as,

$$c(X) = a(X) * b(X)$$

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

In other notation, if we write the polynomials a, b and c as coefficient vectors, then, the result $c = a * b$ simply equals to convolution product of two vectors as shown below [7]:

	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

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. NTRU with $N = 263$ provides an equivalent security level of 1024-bit RSA and 160 bits long ECC [5, 10].

Key Generation: To generate key sets of NTRU, one must first choose two random polynomials $f \in L_f$ and $g \in L_g$. These two polynomials must be small polynomials which means their coefficients must be much smaller than q. Besides, $f_q \equiv f^{-1}$ (mod q) and $f_p \equiv f^{-1}$ (mod p) must exist. Then, the following computation is performed, $h \equiv f_q * g$ (mod q). Now h is our public key while f and f_p are our secret keys. For, more information about parameter selection, small polynomials and finding inverses of polynomials we refer to [4, 8, 9].

Encryption: Firstly, a message m which is a small polynomial is chosen from the plaintext set L_m and then a small random polynomial r is chosen from the set L_r as blinding value. Finally, encrypted text is evaluated as, $e \equiv pr * h + m$ (mod q). During encryption, h will be always multiplied by p. So, to avoid unnecessary computation it is suggested to use $h \equiv pf_q * g$ (mod q) [11].

Decryption: In order to decrypt the encrypted text e, one must first compute $a \equiv f * e \equiv pr * g + f * m$ (mod q). 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, $b \equiv a$ (mod p) $= f * m$ should be calculated. As the final step, message can be recovered by the multiplication of f_p and b modulo p, $c \equiv b * f_p$ (mod p) $= m$.

4. Low power design methods

There are many power reduction methods available for both technological and architectural level. Here, we will mention only the techniques we have used.

The circuit is planned to be running at low frequencies (500 kHz) thus, we have used a low leakage technology library for synthesizing our design, which enabled us to obtain considerably lower static power consumptions. At the same time, we have also used certain methods to decrease the dynamic power consumption such as clock gating, operand isolation and precomputation.

The aim of clock gating is to control the clock input of the registers by an enable signal and isolating the clock input unless new values are loaded to registers and decreasing the switching activity by this way. In clock gating circuit [13], it is preferred to use a latch in order to disable glitches which may occur in the clock inputs of registers. Figure 1 shows a clock gated register.

Operand isolation is another way of reducing the switching activity by isolating the input of complex combinational

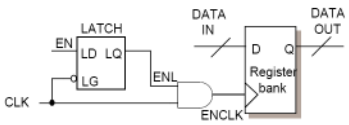

Figure 1. A gated register

circuits (i.e. multipliers, adders) when their outputs are not used.

To avoid consuming power for key generation, public key $h \equiv r * pf_q * g \,(\text{mod q})$ is precomputed and stored in the circuit, since we assume that h remains same at each encryption.

5. Implementation of encryption-only NTRU

We have chosen $N = 167$, $q = 128$ and $p = 3$ among the parameter sets of NTRU that yields to a high level of security [4]. As a result of this choice public key h has coefficients in the interval $[0, 127]$, random polynomial r and message m have coefficients from the set $\{-1, 0, 1\}$. Thus coefficients of h are encoded in 7 bits, and coefficients of r and m are encoded in 2 bits. And, while dealing with negative numbers, two's complement representation is used.

The main architecture of the encryption engine, as shown in Figure 2, consists of a look-up table to store h, a polynomial multiplier (PM) to perform the star multiplication, a Partially Rotating Register (PRR) ($N \times 2$ bits wide) to hold and rotate r and a control logic that manages the whole encryption process.

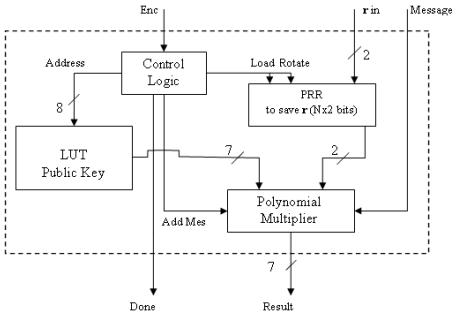

Figure 2. NTRU encryption architecture

In this design we do not consider generation of random polynomials, so we assume that we are able to receive random polynomial coefficients one by one from the input r_{in}.

5.1. Look-up table

The Look-up table (LUT) is implemented in a completely combinational way, in order to give out the appropriate public key coefficients according to its address input.

5.2 Polynomial multiplier

This is the main arithmetic core 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 3.

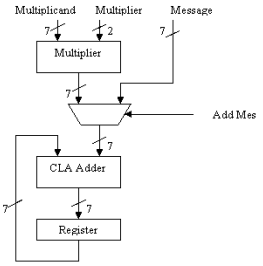

Figure 3. Polynomial multiplier unit

All the submodules of PM are realized for 7 bit computations since we need to reduce the results modulo 128. This way, intermediate and final results are directly reduced during computations.

Multiplier has such a simple structure owing to small coefficients of random polynomial r. Since the coefficients of r take values from the set $\{-1, 0, 1\}$, the product for a multiplicand x is easily computed from $\{-x, 0, x\}$. In case of negative multiplicand, the product is generated by inverting the bits and making the carry input of the CLA adder 1.

CLA adder is chosen because of its non-carry propagation property which we need for less switching activity. At the same time, not to increase the area, the CLA is implemented in two blocks. Intermediate products are saved in the register, and partial sum value is always fed back to one input of CLA adder. After all multiplication and accumulation steps, message input is connected to one input of the adder and final result is generated.

5.3 Partially rotating register

Partially rotating register is a modified version of regular rotating register. We made a modification to benefit from clock gating maximally. During encryption, random polynomial coefficient at the input of PM is changing for every new public key value. So in case of a regular register, whole bits must be shifted one time, which means switching activity of $N \times 2$ registers for every partial product computation. We have seen by our measurements that clock gating a regular rotate register does not have a positive effect on power consumption during NTRU encryption. On the contrary, it increases the dynamic power consumption due to extra switching activity comes from gating circuits (see Table 1). To overcome that negative situation we designed a partially rotating register, as illustrated in Figure 4.

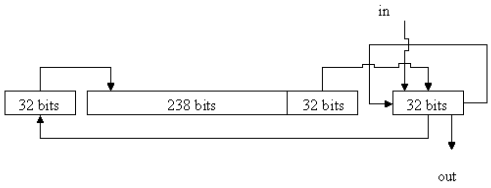

Figure 4. Partially rotating register

In this architecture the right hand side 32-bit register is the part that is always rotating during encryption and loading of random polynomial. During the loading stage most significant two bits of that register are used as input and the output is always the least significant two bits of that register. A *partial rotate* signal is sent by the controller till all 16 values of the register have been used and then a *whole rotate* signal is sent to renew the values of partial rotate register. After receiving that signal, circuit makes a block rotation with width of 32 bits. With this method, only 32 bits are switching constantly during encryption while the rest of the registers switch only 9 times, in the computation of each cipher text word.

5.4 Control logic

Controller of the encryption engine is designed with a finite state machine (FSM) which has four states. It controls the whole process by two 8-bit counters and one 4-bit counter. It starts with the *initial* state after reset and always checks the *Enc* input. If it detects a high signal it transits to the *loading* state. During *loading* state, the coefficients of r are loaded to the PRR one by one. After loading all coefficients, FSM passes to the *multiplication* state.

For *multiplication*, the coefficients of h and r are multiplied and accumulated. *Multiplication* is followed by the *add message* state where message is added to the current sum, cipher text is given out and a done signal is made 1 for one clock cycle. After addition of the message it again transits to *multiplication* state to compute the next coefficient unless it is the last coefficient. After calculating last coefficient it transits to the *initial* state.

6 Implementation of encryption-decryption NTRU

The same parameter sets are used also for encryption-decryption NTRU design. In this case, we have two more polynomials that are, private keys f and f_p. f has coefficients from the set $\{-1, 0, 1\}$ while f_p has coefficients from the set $\{0, 1, 2\}$.

Figure 5 shows the architecture of the design. For simplicity not all of the modules and inner signals are shown. For decryption, a Mod-3 unit, two more look-up tables to store private keys f and f_p, a $N \times 7$ bit PRR and a result register is added to the previous design.

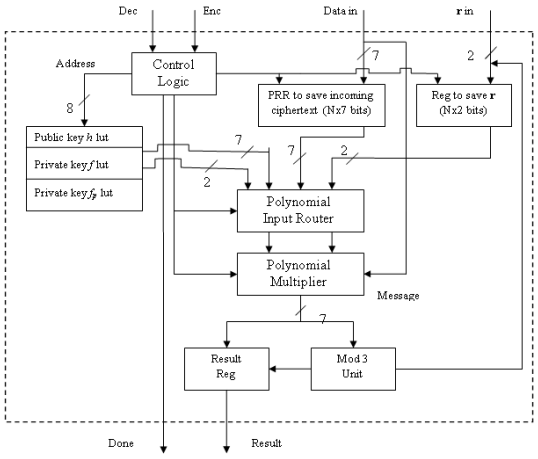

Figure 5. NTRU encryption decryption architecture

Moreover there are four routers, composed by multiplexers, which direct the correct values to the correct inputs according to encryption and decryption stage. During encryption, 7-bit and 2-bit input of PM are bounded to LUT h and $N \times 2$ wide register outputs, respectively. Output of PM is connected to the result register. In the first multiplication of decryption, 7-bit and 2-bit input of PM are bounded to $N \times 7$ wide PRR and LUT f, respectively. And output of PM is connected to Mod-3 unit, while the output of Mod-3 unit is connected to the input of register $N \times 2$. During second multiplication of decryption, 7-bit and 2-bit input of PM are connected to LUT f_p and register $N \times 2$, respectively. Output of PM is again connected to Mod-3 unit but the output of Mod-3 unit is bounded to result register.

The components used in the NTRU encryption engine are also used in this design. Again, we used look-up tables to store the public key and private keys. $N \times 7$ bits wide PRR has the same architecture as defined in Sect. 5, only with modified bit lengths. $N \times 2$ bits wide register is implemented as a regular left rotating register. Result register is also a regular register with a load signal.

The only change has occurred in PM. Since, we designed PM only working in modulo 128, we added an $overflow$ output, which is simply carry output of the CLA, to count the number of modulo 128 reductions during second multiplication of decryption which the computations should be done modulo 3. Also, in this design PM is capable of multiplying with 2 which is used during decryption process.

6.1 Routers

There are four routers in the design. One for address input of the look-up tables, one for input of the PM, one

for rotating register inputs, and one for result register input. These modules maintain the necessary connections between modules according to the control signals generated by controller.

6.2 Mod-3 Unit

Reduction mod 3 is done by a finite state machine based circuit [12]. The FSM starts at state 0 and checks the bits from left to right and makes state transitions according to the value of that bit. The final state after checking the last bit is the value of the number in modulo 3.

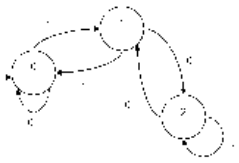

Figure 6. Mod-3 FSM

Overflow output of PM is also connected to this module. There is a 2-bit overflow counter that counts the occurrences of overflow. By using the value of this counter, we obtain the correct reduction mod 3 results.

6.3 Control logic

Controller of the design is implemented as an FSM that has seven states. It controls the state transitions with two 8-bit counters and one 4-bit counter. The FSM starts with the *initial* state following up the reset and directly transits to the *checking* state by controlling the *Enc* and *Dec* inputs. *Enc* and *Dec* inputs are checked continuously at *checking* state. On detection of a high signal at one of these inputs, FSM transits to *loading* state, otherwise state remains unchanged.

During *loading* state, if it is encryption, coefficients of r are loaded to the register which is $N \times 2$ wide, one by one. If it is decryption, coefficients of cipher text e are loaded to the $N \times 7$ wide PRR, one by one. *Loading* is followed by the *multiplication* state. *Multiplication* state is common for encryption and decryption, since polynomial multiplication is carried out in the same way for both situations. After finishing the multiplication and accumulation of one coefficient it transits to *add message* state during encryption otherwise transits to *reduction 3* state.

During *add message* state, message is added to the current sum and controller goes to *final result* state. 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. Both for the first and second reduction mod 3, controller goes to *final result* state after computation.

At *final result* state, if the circuit is doing encryption, controller loads the result to result register and makes the *done* output 1 for one clock cycle. If decryption is being carried out and circuit is performing the first multiplication, FSM loads the reduction 3 result to $N \times 2$ wide register. If the second multiplication is being carried out, then reduction 3 result is loaded into result register and *done* output is made 1 for one clock cycle. For all situations if the last coefficient is given out, system transits to *initial* state, otherwise, it transits to *multiplication* state.

7 Analysis

We wrote our designs in GEZEL [14] and optimized them for low power and low area. As the technology library, the Faraday Low Leakage $0.13\,\mu m$ library is used. Designs are synthesized by Synopsys Design Vision at $500\,kHz$ and average power measurements are done by Synopsys Power Compiler. All power measurements are done by using the switching activity which is captured by gate-level simulation with ModelSim. Since, the most detailed low-cost NTRU implementation belongs to the work of Kaps, we compare our results with his work [6].

We synthesized three different designs for encryption. One is without any enhancements for power consumption, one is with clock gated registers, and the last one is with clock gated registers and partially rotating registers. We will denote them as *Enc1, Enc2* and *Enc3* respectively. Gate counts of the designs found by dividing the whole area of the circuit by the area of one 2-input NAND gate.

	Area			Power		
	Comb	Non-comb	Total	P_{dyn} (μW)	P_{sta} (nW)	P_{tot} (μW)
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
[6]	523	2,327	2,850	4.03	$15.1e+03$	19.3

Table 1. Power and area results of only encryption NTRU

As we can see from Table 1, our most optimized design *Enc3* and the previous work have almost same gate numbers whereas our design is superior to [6], with $1.72\,\mu W$ dynamic and $1.74\,\mu W$ total power consumption. Owing to technology library that has pW as the leakage power unit, we also measured considerably lower static power consumptions. Since the rotating register is the major source of area (73.6%) and power (82.6 %) consumption in *Enc1*, using partially rotating registers enables us to obtain dynamic power savings of more than 50 %. Furthermore, total power consumption of the *Enc3* design during idle state is measured as $0.18\,\mu W$.

Our design finishes encryption at $N \times (N+1) + N$ clock cycles. For our case $N = 167$ and with a clock frequency of $500\,kHz$, it takes $56.44\,ms$ (28 223 clock cycles), which is 3.5% faster than [6] that finishes encryption at $58.45\,ms$ (29 225 clock cycles).

To the best of our knowledge, no encryption-decryption NTRU design is reported before, so we only give our results. Table 2 shows the consumption of optimized design. As seen from the table total gate count is 10.5 kgates and 84% of the area is occupied by registers.

Block	Comb	Non-comb	Total	%
Controller	231	172	403	3.8
Luts	717	0	717	6.8
PM	109	81	190	1.8
Reduction 3	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

Table 2. Area consumption of encryption-decryption NTRU

Power consumption of the circuit is measured for three different working states of the design: Encryption, decryption and idle. Table 3 shows these results.

As seen our optimized design consumes only about $6\,\mu W$ while encryption and decryption and $0.5\,\mu W$ during idle state. Rotating registers are the major sources that consume power. Around 80% of the power is consumed by these registers during encryption and decryption.

For the encryption-decryption NTRU design, encryption takes $N \times (N + 2) + N$ clock cycles while decryption takes $2 \times N \times (N + 11) + N$ clock cycles. For the case $N = 167$ and with a clock frequency of $500\,kHz$, encryption takes $56.78\,ms$ (28 390 clock cycles), decryption takes $119.23\,ms$ (59 619 clock cycles).

8 Conclusion

In this paper we presented a compact and low power NTRU design that is suitable for pervasive security applications such as RFIDs and sensor nodes. We designed two architectures. One is only capable of encryption and the other one performs both, encryption and decryption. Our encryption-only NTRU design has a gate-count of 2.8 kgates, which makes it a very compact solution. The encryption takes $56.44\,ms$ and this makes it 3.5% faster than the best previous work. Our design consumes $1.72\,\mu W$ of dynamic power. 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.We obtained a gate count of 12.3 kgates for the design, which we further improved to 10.5 kgates. The optimized design has the following results, $5.93\,\mu W$, $6.04\,\mu W$ and $0.45\,\mu W$ for dynamic power consumption during encryption, decryption and idle state, respectively. The latency for encryption and decryption, takes $56.78\,ms$ and $119.23\,ms$ respectively.

		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

Table 3. Power results of encryption decryption NTRU

Furthermore, in order to speed up the designs more PM units may be used in parallel with a little change in the rest of the design. Also, it is possible to reduce the multiplication number needed for decryption from 2 to 1, by selecting algorithm parameters in a different way.

References

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

[2] L. Batina, N. Mentens, K. Sakiyama, B. Preneel, and I. Verbauwhede. Low-cost elliptic curve cryptography for wireless sensor networks. In *4th European Workshop on Security and Privacy in Ad hoc and Sensor Networks, Lecture Notes in Computer Science*, volume 4537, pages 6–17. Springer-Verlag, 2006.

[3] G. Gaubatz, J.-P. Kaps, E. Öztürk, and B. Sunar. State of the art in ultra-low power public key cryptography for wireless sensor networks. In *Third IEEE Int. Conf. Pervasive Comput. Commun. Workshops*, volume v2005, pages 146–150. IEEE Computer Society, Mar 2005.

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

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

[6] J. Kaps. *Cryptography for Ultra-Low Power Devices*. PhD thesis, Worcester Polytechnic Institute, May 2006.

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

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

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

[10] http://www.ntru.com/cryptolab/faqs.htm#sixteen.

[11] The NTRU Public Key Cryptosystem A-Tutorial.

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

[13] Synopsys, Inc. *Power Compiler User Guide*, 2006.

[14] http://rijndael.ece.vt.edu/gezel2/index.php.