

ECRYPT II



ICT-2007-216676

ECRYPT II

European Network of Excellence in Cryptology II

Network of Excellence

Information and Communication Technologies

D.VAM.2

Report on “Lightweight Cryptography”

Due date of deliverable: January 2010 (extended to July 2010).

Actual submission date: 8. July 2010

Start date of project: 1 August 2008

Duration: 4 years

Lead contractor: Katholieke Universiteit Leuven (KUL)

Revision 1.0

Project co-funded by the European Commission within the 7th Framework Programme		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission services)	
RE	Restricted to a group specified by the consortium (including the Commission services)	
CO	Confidential, only for members of the consortium (including the Commission services)	

Report on “Lightweight Cryptography”

Editor

Ingrid Verbauwhede (KUL)

Contributors

Ali Can Atici (KUL)

Lejla Batina (KUL)

Andrey Bogdanov (RUB)

Christophe De Cannière (KUL)

Orr Dunkelman (KUL)

Junfeng Fan (KUL)

Miroslav Knežević (KUL)

Gregor Leander (RUB)

Christof Paar (RUB)

Axel Poschman (RUB, NTU)

Kai Schramm (RUB)

Ingrid Verbauwhede (KUL)

8. July 2010

Revision 1.0

The work described in this report has in part been supported by the Commission of the European Communities through the ICT program under contract ICT-2007-216676. The information in this document is provided as is, and no warranty is given or implied that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

Contents

1	Symmetric Key Cryptography	1
1.1	Introduction	1
1.2	DESL and DESXL	1
1.2.1	Introduction	1
1.2.2	Implementation Results	1
1.3	PRESENT	1
1.3.1	Introduction	1
1.3.2	Implementation Results	2
1.4	KATAN & KTANTAN	2
1.4.1	Introduction	2
1.4.2	Implementation Results	3
2	Public Key Cryptography	3
2.1	Introduction	3
2.2	Elliptic Curve Based Security Processor for RFID	4
2.2.1	Introduction	4
2.2.2	Implementation Results	4
2.3	Lightweight Implementation of HECC	6
2.3.1	Introduction	6
2.3.2	Implementation Results	6
2.4	Low-cost Implementations of NTRU for Pervasive Security	7
2.4.1	Introduction	7
2.4.2	Implementation Results	8

1 Symmetric Key Cryptography

1.1 Introduction

Low-end devices, such as RFID tags, are deployed in increasing numbers each and every day. Such devices are used in many applications and environments, leading to an ever increasing need to provide security (and privacy). The problem of providing secure primitives in these devices is the extremely constrained environment. The primitive has to have a small footprint (where any additional gate might lead to the solution not being used), reduced power consumption (as these devices either rely on a battery or on an external electromagnetic field to supply them the required energy), and with sufficient speed (to allow the use of the primitive in real protocols). The raising importance as well as the lack of secure and suitable candidates, has initiated a research aiming to satisfy these requirements. Several suitable building blocks, such as secure block ciphers, have been developed within the ECRYPT II project, VAMPIRE-WG1.

The first candidate block cipher for these devices is the DESL algorithm [17], and its security stronger variant DESXL. DESL is based on the general structure of DES, while using a specially selected S-box. DESL has key size of 56 bits and a footprint of 1,848 GE. The second candidate is the PRESENT block cipher [5]. PRESENT has an SP-Network structure, and it can be implemented using the equivalent of 1,570 GE. A more dedicated implementation of PRESENT in $0.18\mu\text{m}$ CMOS technology reaches 1,075 GE [25]. Finally, the third candidate for the mission is a family of KATAN & KTANTAN block ciphers [6]. It is a light-weight block cipher that is based on a simple structure of two non-linear feedback shift registers coupled with light quadratic Boolean functions. The smallest variant of the family with a fixed key and a block size of 32 bits (KTANTAN32) has a size of only 462 GE.

1.2 DESL and DESXL

1.2.1 Introduction

DESXL was proposed by Leander *et al.* in [17] and is based on DESL, a modified variant of DES. DESL is similar to DES except for the substitution layer, where the eight S-boxes are replaced with a single S-box that is repeated eight times. Furthermore the Initial Permutation and its inverse (IP, IP^{-1}) are omitted in DESL. Additionally DESXL uses key-whitening techniques to increase the key length.

1.2.2 Implementation Results

Poschmann *et al.* describe lightweight hardware implementations of DESL and DESXL in [24]. Their architecture uses a serialized data path and requires 144 clock cycles to encrypt one data block. On a 180 nm technology their implementation of DESL requires 1,848 GE while DESXL requires 2,168 GE. Table 1 summarizes their results.

1.3 PRESENT

1.3.1 Introduction

PRESENT was proposed in 2007 [5]. It is a Substitution Permutation Network (SPN) with a block size of 64 bits, 31 rounds and two key sizes, 80 and 128 bits.

Table 1: Hardware implementation results of DES and DESXL. All figures are obtained at or calculated for a frequency of 100KHz. Please be aware that power figures can not be compared adequately between different technologies.

Alg.	key size	block size	datapath width	Enc. / Dec.	cycles / block	T'put [Kbps]	Tech. [μm]	Area [GE]	Eff. [bps/GE]	Cur. [μA]	Ref.
DESL	56	64	4	Enc	144	44.44	0.18	1,848	24.05	0.89	[24]
DESXL	184	64	4	Enc	144	44.44	0.18	2,168	20.5	N/A	[24]

1.3.2 Implementation Results

The designers of PRESENT report encryption-only ASIC implementations in [5]. An area-optimized round-based PRESENT-80 implementation requires 1,570 GE and 32 clock cycles to perform a single encryption. A low-power implementation with a similar architecture that trades area for power is requires 1,623 GE. PRESENT-80 needs 32 clock cycles and 1,886 GE.

Rolfes *et al.* report several hardware implementations of PRESENT in [25]. Using a serialized architecture PRESENT-80 requires only 1,075 GE and 547 clock cycles. Details for round-based and serialized hardware implementations of PRESENT-128 and a summary for a wide variety of different PRESENT implementations can be found in [23]. Table 2 summarizes the implementation results of PRESENT.

Table 2: Hardware implementation results of PRESENT. All figures are obtained at or calculated for a frequency of 100KHz. Please be aware that power figures can not be compared adequately between different technologies.

key size	block size	datapath width	Enc. / Dec.	cycles / block	T'put [Kbps]	Tech. [μm]	Area [GE]	Eff. [bps/GE]	Cur. [μA]	Ref.
80	64	4	Enc	547	11.7	0.18	1,075	10.89	1.4	[25]
80	64	64	Enc	32	200	0.18	1,570	127.4	2.78	[5]
80	64	64	Enc	32	200	0.18	1,623	127.4	1.83	[5]
128	64	4	Enc	559	11.45	0.18	1,391	8.23	N/A	[23]
128	64	64	Enc	32	200	0.18	1,884	106.2	3.67	[23]

1.4 KATAN & KTANTAN

1.4.1 Introduction

A family of KATAN & KTANTAN is a family small and efficient, hardware-oriented block ciphers. It was published by De Cannière *et al.* [6] in 2009. The family contains six block ciphers divided into two flavors. All block ciphers share the 80-bit key size and security level. The first flavor, KATAN, is composed of three block ciphers, with 32, 48, or 64-bit block size. The second flavor, KTANTAN, contains the other three ciphers with the same block sizes, and is more compact in hardware, as the key is burnt into the device (and cannot be changed).

1.4.2 Implementation Results

The authors of KATAN & KTANTAN have reported encryption-only ASIC implementations of the whole family [6]. The smallest cipher of the entire family, KTANTAN32, can be implemented in 462 GE while achieving encryption speed of 12.5 Kbps (at 100 KHz). KTANTAN48, which is the version authors recommend for RFID tags uses 588 GE, whereas KATAN64, the largest and most flexible candidate of the family, uses 1054 GE and has a throughput of 25.1 Kbps (at 100 KHz). Table 1.4.2 summarizes the implementation results of KATAN & KTANTAN.

Table 3: Area-Throughput Trade-Offs.

Cipher	Block (bits)	Key (bits)	Size (GE)	GE per Memory Bit	Throughput* (Kb/s)	Logic Process
KATAN32	32	80	802	6.25	12.5	0.13 μm
KATAN32	32	80	846	6.25	25	0.13 μm
KATAN32	32	80	898	6.25	37.5	0.13 μm
KATAN48 [†]	48	80	916	6.25	9.4	0.13 μm
KATAN48	48	80	927	6.25	18.8	0.13 μm
KATAN48	48	80	1002	6.25	37.6	0.13 μm
KATAN48	48	80	1080	6.25	56.4	0.13 μm
KATAN64 [†]	64	80	1027	6.25	8.4	0.13 μm
KATAN64	64	80	1054	6.25	25.1	0.13 μm
KATAN64	64	80	1189	6.25	50.2	0.13 μm
KATAN64	64	80	1269	6.25	75.3	0.13 μm
KTANTAN32	32	80	462	6.25	12.5	0.13 μm
KTANTAN32	32	80	673	6.25	25	0.13 μm
KTANTAN32	32	80	890	6.25	37.5	0.13 μm
KTANTAN48 [†]	48	80	571	6.25	9.4	0.13 μm
KTANTAN48	48	80	588	6.25	18.8	0.13 μm
KTANTAN48	48	80	827	6.25	37.6	0.13 μm
KTANTAN48	48	80	1070	6.25	56.4	0.13 μm
KTANTAN64 [†]	64	80	684	6.25	8.4	0.13 μm
KTANTAN64	64	80	688	6.25	25.1	0.13 μm
KTANTAN64	64	80	927	6.25	50.2	0.13 μm
KTANTAN64	64	80	1168	6.25	75.3	0.13 μm

* — A throughput is estimated for frequency of 100 KHz.

† — Using clock gating.

2 Public Key Cryptography

2.1 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 and *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 [14].

Several PKC implementations have been reported for passive RFID tags. For instance, Elliptic Curve Cryptosystem (ECC) was implemented for RFID tags [15, 18]. According to [1], a passive RFID tag should have power consumption less than $15 \mu\text{W}$ to guarantee a 1 meter operation range. Some ECC implementations can already fulfill the requirements. For example, ECC processor proposed by Lee *et al.* [18], using 15.4 kGE with 130 nm technology, consumes $12.08 \mu\text{W}$ when running at 323 kHz, and one scalar multiplication takes only 243 *ms*. The ECC core proposed by Hein *et al.* [15] consumes $10.08 \mu\text{W}$ when running at 106 kHz.

2.2 Elliptic Curve Based Security Processor for RFID

2.2.1 Introduction

An architecture of a state-of-the-art processor for RFID tags with an Elliptic Curve (EC) processor over $\text{GF}(2^{163})$ was reported in [18]. The plausibility of meeting both security and efficiency requirements even in a passive RFID tag was demonstrated. The proposed processor is able to perform EC scalar multiplications as well as general modular arithmetic (additions and multiplications) which are needed for the cryptographic protocols. To obtain an efficient modulo arithmetic, a redundant modular operation was introduced. Moreover, the proposed architecture can support multiple cryptographic protocols. The synthesis results with a $0.13 \mu\text{m}$ CMOS technology show that the gate area of the most compact version is only 12.5 kGE.

Elliptic Curve Cryptography (ECC) includes protocols that are based on arithmetic of elliptic curves. Curves that are commonly used in applications are usually defined over $\text{GF}(p)$ or $\text{GF}(2^n)$, where p is a prime number. Elliptic curve systems over both types of fields provide the same level of security but the so-called binary fields have some implementation advantages. Namely, binary arithmetic is “carry-free”, squaring can be implemented very efficiently in some cases *etc.* The properties are very convenient for hardware implementations. Binary fields offer also more arithmetic options as there are many choices for bases, irreducible polynomials, fields, *etc.* In general, the elliptic curve arithmetic consists of several hierarchical levels. The top level is EC scalar multiplication which is executed by point addition and doubling. The point operations can be performed by different formulae, which depend on the representation chosen *i.e.* coordinates. The formulae for point arithmetic are sequences of finite field operations: addition/subtraction, multiplication/squaring and inversion.

2.2.2 Implementation Results

In order to find the best tradeoffs, the authors have proposed three different architectures of the ECP as shown in Table 4. Type 1 is the minimal version described so far. Type 2 uses an extra register to hold the X -coordinate value of the base EC point (*i.e.* P at the EC scalar multiplication of $k \cdot P$), say $x(P)$. Therefore, this extra register makes the ECP load $x(P)$ only once and use for the whole calculation of an EC scalar multiplication. Otherwise, the ECP has to load $x(P)$ at every iteration in the Montgomery algorithm, which means that the ECP has to load 163 times for a 163-bit key. Type 3 has an extra register and a randomly accessible register file. The use of the extra register and the randomly accessible register file increase the gate area while reduce the number of cycles. Therefore, the ECP Type 1 has the least gate area and the most number of cycles and Type 3 has the most gate area and the least number of cycles in the same digit size.

Table 4: EC Processor Types.

	An extra buffer register	The register file type
Type 1	No	Circular Shift Register File
Type 2	Yes	Circular Shift Register File
Type 3	Yes	Randomly Accessible Register File

The proposed architectures are synthesized using a low leakage power library of UMC 0.13 μm (fsc0l_d_sc_tc.db). The synthesized architectures include the micro controller, the bus manager and the ECP. Some samples of the synthesis results and the performances are shown in Table 5. The number of cycles is to finish the Schnorr protocol, which includes one EC scalar multiplication, some general modular operations, the random number generation and the data transmission/reception.

The gate area is dominated by the register file. In order to minimize the gate area, the authors minimize the flip-flops. The UMC standard cell library of 0.13 μm offers a very compact D flipflop combined with a multiplexer, which can be implemented in 6.25 GE. In the case of Type 1 and the digit size of 1, the register file occupies 7.53 KGE. This is around 7.7 GE per bit including the multiplexers.

Table 5: Synthesis Results and Performance.

Type	Digit Size	ECP Gate Area	Overall ⁽¹⁾	Cycles ⁽²⁾ Gate Area	Frequency (kHz)	Time ⁽²⁾ (msec)	Dynamic Power (μW)	Leakage Power (μW)	Total Power (μW)
1	1	10,106	12,506	302,457	1,130	244.43	36.5780	0.0509	36.6289
	2	11,383	14,064	171,480	590	246.33	21.4927	0.0553	21.5480
	3	12,236	14,729	127,821	411	247.18	15.6854	0.0609	15.7463
	4	12,863	15,356	105,183	323	244.47	12.0117	0.0641	12.0758
	5	13,497	15,989	92,247	266	248.20	11.3389	0.0674	11.4063
2	1	11,133	13,624	298,111	1,130	240.58	38.6624	0.0559	38.7183
	2	12,696	15,191	167,136	565	249.35	22.5107	0.0622	22.5729
	3	13,319	15,808	123,475	399	243.77	16.0403	0.0654	16.1057
	4	13,934	16,433	100,837	301	247.51	12.7105	0.0686	12.7791
	5	14,570	17,251	87,901	251	245.50	11.0944	0.0721	11.1665
3	1	14,307	16,799	293,587	1,130	236.58	43.3769	0.0673	43.4442
	2	15,967	18,451	162,608	565	241.33	24.0127	0.0751	24.0878
	3	16,568	19,074	118,951	377	246.10	16.7974	0.0783	16.8757
	4	17,200	19,693	96,311	283	247.99	13.0870	0.0814	13.1684
	5	17,837	20,316	83,375	230	248.54	11.1542	0.0845	11.2387

⁽¹⁾ The synthesis results are for RFID processor which includes the micro controller, the bus manager and ECP.

⁽²⁾ The number of cycles and the time needed to complete the Schnorr protocol.

2.3 Lightweight Implementation of HECC

2.3.1 Introduction

Hyperelliptic curves are a special class of algebraic curves; they can be viewed as a generalization of elliptic curves. Namely, a hyperelliptic curve of genus $g = 1$ is an elliptic curve, while in general, hyperelliptic curves can be of any genus $g \geq 1$.

Let $\overline{\text{GF}}(2^m)$ be an algebraic closure of the field $\text{GF}(2^m)$. Here we consider a hyperelliptic curve C of genus $g = 2$ over $\text{GF}(2^m)$, which is given with an equation of the form:

$$C : y^2 + h(x)y = f(x) \quad \text{in} \quad \text{GF}(2^m)[x, y], \quad (1)$$

where $h(x) \in \text{GF}(2^m)[x]$ is a polynomial of degree at most g ($\deg(h) \leq g$) and $f(x)$ is a monic polynomial of degree $2g + 1$ ($\deg(f) = 2g + 1$). Also, there are no solutions $(x, y) \in \overline{\text{GF}}(2^m) \times \overline{\text{GF}}(2^m)$ which simultaneously satisfy the equation (1) and the equations: $2v + h(u) = 0, h'(u)v - f'(u) = 0$. These points are called singular points. For the genus 2, in the general case the following equation is used $y^2 + (h_2x^2 + h_1x + h_0)y = x^5 + f_4x^4 + f_3x^3 + f_2x^2 + f_1x + f_0$.

A divisor D is a formal sum of points on the hyperelliptic curve C i.e. $D = \sum m_P P$ and its degree is $\deg D = \sum m_P$. Let Div denote the group of all divisors on C and Div_0 the subgroup of Div of all divisors with degree zero. The Jacobian J of the curve C is defined as quotient group $J = \text{Div}_0 / P$. Here P is the set of all principal divisors, where a divisor D is called principal if $D = \text{div}(f)$, for some element f of the function field of C ($\text{div}(f) = \sum_{P \in C} \text{ord}_P(f) P$). The discrete logarithm problem in the Jacobian is the basis of security for HECC. In practice, the Mumford representation according to which each divisor is represented as a pair of polynomials $[u, v]$ is usually used. Here, u is monic of degree 2, $\deg(v) < \deg(u)$ and $u|f - hv - v^2$ (so-called reduced divisors). For implementations of HECC, we need to implement the multiplication of elements of the Jacobian i.e. divisors with some scalar.

2.3.2 Implementation Results

An implementation of the HECC processor, using 130 nm standard cell CMOS library, was reported in [13]. Table 6 summarizes the area and power of the proposed design.

The reported HECC implementation uses 14.5 kGE and finishes one divisor multiplication in 136838 clock cycles. The power consumption, estimated with power compiler, is around 13.4 μW when running at 300 kHz. The implementation of [26], using projective coordinates, requires 266133 clock cycles for one scalar multiplication. Note that the implementation in [26] is defined on a smaller field and the result does not include data memory. The power and energy consumption of the reported design is also significantly lower when achieving the same delay compared to that of [26]. The main speedup comes from the fast inverter and the use of affine coordinates.

There are many ECC implementations proposed for RFID tags. Lee *et al.* [18] explored the trade-off between area and power of ECC implementation using digit-serial multiplier with various digit size. From $d = 2$ to $d = 4$, the number of clock cycles for one scalar multiplication is halved, while the area increases only 10 %. A trade-off can be made between power, performance and energy.

The architecture proposed in [15] uses a different approach. It utilizes 16x16 GF(2) multiplier and 32-bit accumulator. With shorter registers, the power consumption can be significantly reduced. On the other hand, it requires 296k clock cycles, twice as many as the reported HECC implementation, for one scalar multiplication, which makes the overall energy requirement significantly higher.

The reported HECC processor can meet the requirements for passive RFID tags in terms of area, power and energy. However, the current implementation can hardly match the energy efficiency of

Table 6: Performance comparison of HECC and ECC implementations targeting RFID tags.

Ref. Design	Platform	Finite Field	Area [kGE]	Perf. [#cycle]	Freq. [kHz]	Power [μ W]	Energy* [μ J]	Comments
HECC This work	130 nm ASIC	GF(2^{83})	14.5	136838	300	13.4	6.03	Unified mult./inv. ($d = 4$)
HECC Sakiyama [26]	130 nm ASIC	GF(2^{67})	7.6 [†]	266133	500	19 [†]	10.0 [†]	1 Mult. ($d = 8$)
ECC Lee <i>et al.</i> [18]	130 nm ASIC	GF(2^{163})	14.1 [‡]	144842	590	21.55	5.29	($d = 2$)
14.7 [‡]			101183	411	15.75	3.88	($d = 3$)	
15.4 [‡]			78544	323	12.08	2.94	($d = 4$)	
ECC Hein <i>et al.</i> [15]	180 nm ASIC	GF(2^{163})	11.9	296000	106	10.8	31.3 [◊]	16x16 mult.

[†] Modular Arithmetic Logic Unit only

[‡] Including ECC core and an 8-bit controller for cryptographic protocols

* Energy for one scalar multiplication

[◊] Estimated by authors

some ECC implementations [18]. This is due to the fact that the computational complexity of HECC divisor scalar multiplication is higher than the point multiplication for ECC. Because divisor operations in HECC are much more complicated than point operations in ECC, almost double number of clock cycles is required for one scalar multiplication even if the multiplier has the same digit size.

Note that the current implementation is based on binary scalar multiplication method. No countermeasures for side-channel attacks are deployed. The ECC implementations [18, 15] use Montgomery scalar multiplication, which is believed to be secure against simple power analysis. Adding countermeasures normally causes area increase, performance degradation or both.

2.4 Low-cost Implementations of NTRU for Pervasive Security

2.4.1 Introduction

NTRU [16] 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 $R = \mathbb{Z}[X]/(X^N - 1)$. 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 an other 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 [4] and referred with the $*$ symbol. Thus, the product of two polynomials a and b

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

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

can be calculated as,

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

$$c_k = a_0b_k + a_1b_{k-1} + \dots + a_{N-1}b_{k+1} = \sum_{i+j=k \bmod N} a_i b_j$$

In an other way, 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 [21]:

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

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 .

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 . Also the polynomial f must have inverses modulo (q) which is f_q and modulo (p) which is f_p . Then, the following computation is performed,

$$h \equiv f_q * g \pmod{q}$$

Now h is the public key while f and f_p are the secret keys. For, more information about parameter selection, small polynomials and finding inverses of polynomials we refer to [16, 27, 28].

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:

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

During encryption, h will be always multiplied by p . So, to avoid unnecessary computation it is suggested to use $h \equiv pf_q * g \pmod{q}$ [11].

Decryption: In order to decrypt the encrypted text e , one must first compute

$$a = f * e \pmod{q} = pr * g + f * m \pmod{q}$$

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

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

should be calculated. As the final step message can be recovered by the multiplication of f_p and b modulo p :

$$c = b * f_p \pmod{p} = m$$

2.4.2 Implementation Results

The low cost implementation of NTRU, optimized for low power and low area, is reported in [3]. Three different designs for encryption were synthesized in $0.13 \mu\text{m}$ CMOS technology. One without

Table 7: Power and area results of only encryption NTRU

	Area			Power		
	Comb (GE)	Non-comb (GE)	Total (GE)	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

any enchantments for power consumption, one with clock gated registers, and the last one with clock gated registers and partially rotating registers (denoted as *Enc1*, *Enc2* and *Enc3*, respectively). Table 7 summarizes the implementation results.

Furthermore, the authors report the implementation result of the NTRU module that supports both encryption and decryption. Table 8 shows the area consumption of optimized design. As seen from the table total gate count is 10.5 kGE and 84 % of the area is occupied by registers.

Table 8: Area consumption of encryption-decryption NTRU

Block	Comb (GE)	Non-comb (GE)	Total (GE)	%
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

Power consumption of the circuit is measured for three different working states of the design: Encryption, decryption and idle. Table 9 shows these results.

Table 9: Power results of encryption-decryption NTRU

		Power		
		P_{dyn} (μW)	P_{sta} (nW)	P_{tot} (μW)
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

The optimized design consumes only about 6 μW during the encryption and decryption and 0.5 μW during the idle state. 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 28 390 cycles (56.78 ms), decryption takes 59 619 cycles (119.23 ms).

References

- [1] ISO/IEC 18000-1:2004, Information Technology – Radio Frequency Identification for Item Management. Part 3: Parameters for Air Interface Communications at 13,56 MHz.
- [2] M. Albrecht and C. Cid. Algebraic Techniques in Differential Cryptanalysis. In Orr Dunkelman, editor, *Fast Software Encryption 2009 – FSE 2009*, volume 5665 of LNCS, pages 193–208. Springer-Verlag, 2009.
- [3] A. C. Atici, L. Batina, J. Fan, I. Verbauwhede, and S. B. Örs. Low-cost Implementations of NTRU for pervasive security. In *19th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP 2008)*, pages 79–84, Leuven, 2008. IEEE.
- [4] D. Bailey, D. Coffin, A. Elbirt, J. Silverman, and A. Woodbury. NTRU in Constrained Devices. *Cryptographic Hardware and Embedded Systems — CHES 2001*, pages 262–272, 2001.
- [5] A. Bogdanov, G. Leander, L. R Knudsen, C. Paar, A. Poschmann, M. J.B Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT - an Ultra-Lightweight Block Cipher. In *Proceedings of CHES 2007*, LNCS, pages 450 – 466. Springer-Verlag, 2007.
- [6] M. Knežević C. D. Cannière, O. Dunkelman. KATAN & KTANTAN — A Family of Small and Efficient Hardware-Oriented Block Ciphers. In *Workshop on Cryptographic Hardware and Embedded Systems — CHES*, pages 272–288, 2009.
- [7] J. Y. Choo. Linear Cryptanalysis of Reduced-Round PRESENT. In *Topics in Cryptology - CT-RSA 2010*, volume 5985 of LNCS, pages 302–317. Springer-Verlag, 2010.
- [8] B. Collard and F.-X. Standaert. A Statistical Saturation Attack against the Block Cipher PRESENT, Errata and Improvement. Available for download via <http://www.dice.ucl.be/~fstandae/PUBLIS/62b.pdf>.
- [9] B. Collard and F.-X. Standaert. A Statistical Saturation Attack against the Block Cipher PRESENT. In *Topics in Cryptology — CT-RSA 2009*, volume 5473 of *Lecture Notes in Computer Science*, pages 195–210. Springer-Verlag, 2009.
- [10] B. Collard and F.X. Standaert. Multi-Trail Statistical Saturation Attacks. In *Proceedings of ACNS 2010*, LNCS. Springer-Verlag, 2010, to appear.
- [11] NTRU Cryptosystems. The NTRU Public Key Cryptosystem — A Tutorial.
- [12] B. Esslinger. CrypTool. Available via www.cryptool.de, 2008.
- [13] J. Fan, L. Batina, and I. Verbauwhede. Light-Weight Implementation Options for Curve-Based Cryptography: HECC is also Ready for RFID. In *International workshop on RFID Security and Cryptography - RISC 2009*, page 6, London,UK, 2009.

- [14] 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 *2nd IEEE International Workshop on Pervasive Computing and Communication Security (PerSec 2005)*, Kauai Island, Hawaii, March 2005.
- [15] D. Hein, J. Wolkerstorfer, and N. Felber. ECC is ready for RFID — A Proof in Silicon. In *Proceedings of 15th Annual Workshop on Selected Areas in Cryptography (SAC 2008)*. Lecture Notes in Computer Science, R. Avanzi, L. Keliher and F. Sica (eds.), Springer-Verlag, 2008.
- [16] J. Hoffstein, J. Pipher, and J. Silverman. NTRU: A Ring-Based Public Key Cryptosystem. *Algorithmic Number Theory*, pages 267–288, 1998.
- [17] G. Leander, C. Paar, A. Poschmann, and K. Schramm. New Lightweight DES Variants. In *Proceedings of Fast Software Encryption 2007 – FSE 2007*, volume 4593 of LNCS, pages 196–210. Springer-Verlag, 2007.
- [18] Y. K. Lee, K. Sakiyama, L. Batina, and I. Verbauwhede. Elliptic-Curve-Based Security Processor for RFID. *IEEE Trans. Comput.*, 57(11):1514–1527, 2008.
- [19] J. Nakahara, P. Sepéhrdad, B. Zhang, and M. Wang. Linear (Hull) and Algebraic Cryptanalysis of the Block Cipher PRESENT. In *Proceedings of CANS'09*, volume 5888 of LNCS, pages 58–75. Springer-Verlag, 2009.
- [20] K. Ohkuma. Weak Keys of Reduced-Round PRESENT for Linear Cryptanalysis. In *Selected Areas in Cryptography: 16th Annual International Workshop, SAC 2009, Calgary, Alberta, Canada, August 13-14, 2009, Revised Selected Papers*, pages 249–265, Berlin, Heidelberg, 2009. Springer-Verlag.
- [21] C. M. O’Rourke. Efficient NTRU Implementations. Master Thesis, Worcester Polytechnic Institute, 2002.
- [22] O. Özen, K. Varici, C. Tezcan, and Ç. Kocair. Lightweight Block Ciphers Revisited: Cryptanalysis of Reduced Round PRESENT and HIGHT. In *Proceedings of the 14th Australasian Conference on Information Security and Privacy*, volume 5594 of LNCS, pages 90–107. Springer-Verlag, 2009.
- [23] A. Poschmann. *Lightweight Cryptography - Cryptographic Engineering for a Pervasive World*. Number 8 in IT Security. Europäischer Universitätsverlag, 2009. Published: Ph.D. Thesis, Ruhr University Bochum.
- [24] A. Poschmann, G. Leander, K. Schramm, and C. Paar. New Lightweight Crypto Algorithms for RFID. In *Proceedings of The IEEE International Symposium on Circuits and Systems 2007 – ISCAS 2007*, pages 1843–1846, 2007.
- [25] C. Rolfes, A. Poschmann, G. Leander, and C. Paar. Ultra-Lightweight implementations for smart devices - security for 1000 gate equivalents. In *Proceedings of the 8th Smart Card Research and Advanced Application IFIP Conference — CARDIS 2008*, volume 5189 of LNCS, pages 89–103. Springer-Verlag, 2008.
- [26] K. Sakiyama. *Secure Design Methodology and Implementation for Embedded Public-key Cryptosystems*. PhD thesis, Katholieke Universiteit Leuven, Belgium, 2007.

- [27] J. H. Silverman. Invertibility in Truncated Polynomial Rings. Technical Reports, NTRU Cryptosystems, 1998.
- [28] J. H. Silverman. Almost Inverses and Fast NTRU Key Creation. Technical Reports, NTRU Cryptosystems, 1999.
- [29] M. Wang. Differential cryptanalysis of Reduced-Round PRESENT. In *Proceedings of AFRICACRYPT 2008*, LNCS, page 40–49. Springer-Verlag, 2008.
- [30] M.R. Z’Aba, H. Raddum, M. Henricksen, and E. Dawson. Bit-Pattern Based Integral Attack. In K. Nyberg, editor, *Fast Software Encryption — FSE 2008*, volume 5086 of *Lecture Notes in Computer Science*, pages 363–381. Springer-Verlag, 2008.