Public Key Cryptography in Sensor Networks—Revisited *

Gunnar Gaubatz, Jens-Peter Kaps, Berk Sunar

Department of Electrical & Computer Engineering
Worcester Polytechnic Institute
100 Institute Road, Worcester, MA 01609, U.S.A.
{gaubatz, kaps, sunar}@wpi.edu

Abstract. The common perception of public key cryptography is that it is complex, slow and power hungry, and as such not at all suitable for use in ultra-low power environments like wireless sensor networks. It is therefore common practice to emulate the asymmetry of traditional public key based cryptographic services through a set of protocols [1] using symmetric key based message authentication codes (MACs). Although the low computational complexity of MACs is advantageous, the protocol layer requires time synchronization between devices on the network and a significant amount of overhead for communication and temporary storage. The requirement for a general purpose CPU to implement these protocols as well as their complexity makes them prone to vulnerabilities and practically eliminates all the advantages of using symmetric key techniques in the first place. In this paper we challenge the basic assumptions about public key cryptography in sensor networks which are based on a traditional software based approach. We propose a custom hardware assisted approach for which we claim that it makes public key cryptography feasible in such environments, provided we use the right selection of algorithms and associated parameters, careful optimization, and low-power design techniques. In order to validate our claim we present proof of concept implementations of two different algorithms—Rabin's Scheme and NtruEncrypt—and analyze their architecture and performance according to various established metrics like power consumption, area, delay, throughput, level of security and energy per bit. Our implementation of NtruEncrypt in ASIC standard cell logic uses no more than $3,000$ gates with an average power consumption of less than $20\,\mu$W. We envision that our public key core would be embedded into a light-weight sensor node architecture.

1 Introduction

Wireless *distributed sensor networks* (DSN) are expected to be used in a wide range of applications, from monitoring wildlife and collecting microclimate data

* This material is based upon work supported by the National Science Foundation under Grants No. ANI-0133297 (NSF CAREER Award) and No. ANI-0112889.

[2–4] to a number of military applications like target tracking [5] and detection of biological or chemical weapons. The current generation of wireless sensor nodes is still relying on batteries as its source of power. The limited lifetime of batteries, however, significantly impedes the usefulness of such devices since maintenance accesses would become necessary whenever the battery is depleted. Furthermore, the intention of having large amounts of tiny nodes scattered over a large area would render maintenance impractical. Next generation sensor nodes will therefore combine ultra-low power circuitry with so-called *power scavengers*, which allow for maintenance-free operation of the nodes. This opens up a whole new range of applications where the nodes can be placed in inaccessible locations.

Power scavengers are devices able to harvest small amounts of energy from ambient sources such as light, heat or vibration. This energy is stored in a capacitor and can be used to power the sensor node either continuously, for small amounts of power, or in intervals if the demand is higher. At least $8\mu W$ of power can be generated using MEMS-based power scavengers, as reported in [6]. Other larger systems are able to generate much more power [7], but these are typically not integrated on-chip with the actual sensor node. It is expected that future MEMS-based scavengers will be able to deliver power up to $20\mu W$ continuously.

Due to the sensitive nature of many of the anticipated applications of DSN, a certain minimum level of secure communication between sensor nodes and base station is required. This includes data confidentiality and integrity. Both can be provided through encryption of the data. One of the biggest problems in using secret key algorithms—apart from their size and scalability issues—is the protection of the sensitive key material. Sensor nodes might be deployed in an untrusted environment, e.g. for military applications. The capture of a single node by an adversary should not jeopardize the integrity of the entire network. In a setting where the sensor nodes send encrypted data to a base station a public key scheme is of great advantage as here each node contains only public key material not private. Previously proposed security protocols such as SNEP and μTESLA [1] provide secure authentication using only symmetric key techniques. In order to provide authentication to insecure nodes μTESLA has to emulate asymmetry through a delayed disclosure of symmetric keys. While Carman et al. acknowledge in [8] that symmetric key techniques are attractive due to their energy efficiency, they also conclude that all symmetric key based key exchange protocols analyzed by them exhibit limitations in their flexibility. The emulation of an asymmetric cryptographic primitive requires that each node is time synchronized with the base station and has key management functions and ample storage. As the symmetric keys are revealed sequentially over time, nodes might have to store multiple messages before they can be authenticated. This broadcast authentication scheme also implies that the keys shared among all nodes need to be updated in regular intervals, requiring broadcasts from the base station to all nodes. As in many settings the base station can not directly communicate with all nodes, these keys need to be forwarded from node to node. This protocol overhead leads to increased energy consumption of the nodes as

keys and key management messages need to be transmitted frequently. Complex key management and high storage requirements for multiple keys and messages put a considerable burden on the power consumption of the nodes. The use of public key cryptography would eliminate the need for complicated protocols and at the same time would also increase the security of the entire system, since only the public key of the base station would have to be embedded into the nodes.

The challenge is to overcome the considerable computational complexity of standard public key encryption algorithms and make public key encryption possible in self powered sensor nodes. Traditional schemes like RSA or ElGamal require considerable amounts of resources which in the past limited their use to large-scale platforms like networked servers and personal computers. Mobile equipment with less computational resources, such as cell phones, Personal Digital Assistants (PDAs) and pagers, therefore uses much more efficient elliptic curve based algorithms such as EC-DH and EC-DSA which execute considerably faster while preserving the same level of security [9]. The operands of EC-cryptosystems are much shorter than those in traditional schemes. Unfortunately the improved computational efficiency of ECC comes at the price of much more complex arithmetic primitives and a large number of temporary operands, whereas RSA or ElGamal require only one single arithmetic primitive and few operands. The heterogenous structure and larger storage requirements of ECC make it less scalable and in effect less attractive for energy efficient low-power implementations.

In this paper we compare two architectures that implement two different types of public key crypto-systems with promising characteristics. The first one, Rabin's Scheme [10], is a specialization of the well known RSA algorithm [11] where the exponent is fixed to the value 2. As with RSA, the security of Rabin's scheme relies on the hard problem of factoring large integers. The second algorithm, NtruEncrypt [12], was introduced in 1996 by Hoffstein, Pipher and Silverman. NtruEncrypt is a public key cryptosystem where security is based on the hardness of the Shortest Vector Problem (SVP) in a very high dimension lattice. It still uses relatively large operands, but it reduces the overall asymptotic complexity of the encryption operation to $O(n^2)$ compared to RSA's $O(n^3)$. In both cases we concentrate on the encryption operation only. The decryption of the sensor data would be performed by the more powerful base station. We analyze the performance of these architectures by means of various established metrics in the field of computer organization, like power consumption, area, delay, throughput and latency. We also include some that are not as commonly encountered, such as level of security and energy per bit encrypted. We demonstrate that ultra-low power implementations of public key cryptography are feasible. Our interest, however, is mainly focused on the computational aspects of the underlying arithmetic primitives and as such we refrain from deeper discussion of protocol issues and the cryptographic services that need to be provided by these systems.

The remainder of the paper is structured as follows. After a brief description of the cryptosystems in Section 2, and an introduction into low-power design

techniques 3, we will focus on their application for implementing the algorithms in Section 4. In Section 5 a brief definition of the metrics of interest is followed by an extensive comparative analysis of our two architectures. The final section concludes our findings and points out directions for future work.

2 Preliminaries

Rabins's Scheme and NtruEncrypt are two very different public key algorithms. In this section we first describe the selection of the algorithm specific parameters to make them comparable. Then we give a brief overview of their function.

2.1 Parameter Selection

In order to better compare these two algorithms of disparate properties and parameter sets, we chose system parameters of both algorithms to offer a closely matching level of security. For definition of this level we refer to the widely recognized definition of equivalent security by Lenstra and Verheul [13]. Amongst others they cover RSA as the principal example for cryptosystems where security is based on the *Integer Factorization Problem*, which is also the basis for Rabin's Scheme. Their analysis, however, does not include a definition of equivalent security for a lattice based scheme like NtruEncrypt. For our purposes we therefore refer to the analysis of Hoffstein, Silverman and Whyte [14], who present a similar evaluation of NtruEncrypt's security level, also in terms of equivalent security.

While in practice certain classes of applications might require a higher level of security than others, we regard our designs simply as a proof of concept and hence chose to implement them at a comparatively low level of security. It should, however, be relatively straightforward to estimate the cost of higher security level implementations based on the analysis that we give at the end of this paper. For Rabin's Scheme we selected an operand size of 512 bits, which according to Lenstra and Verheul [13] provides a security level of around 60 bits. In the case of NtruEncrypt we chose the system parameters as $(N, p, q) = (167, 3, 128)$, based on findings in [14], offering a security level of 57 bits.

2.2 Rabin's Scheme

Rabin's Scheme was introduced in 1979 in [10]. It is based on the factorization problem of large numbers and is therefore similar to the security of RSA with the same sized modulus. Rabin's Scheme has asymmetric computational cost. The encryption operation is extremely fast, however decryption times are comparable to RSA of the same modulus. This asymmetry makes Rabin's Scheme especially interesting for our application. Here is a brief description of the Rabin's Scheme. For a more detailed description and the mathematical proofs see [10][15].

Key Generation

1. Choose two large random strong prime numbers.
2. Compute $n = p \cdot q$.
3. Pick a random number b for which $0 \leq b < n$.
4. The public key is (n, b), the private key is (p, q).

Encryption

1. Represent the message as an integer x for which $0 \leq x < n$
2. Compute the ciphertext $\mathrm{E}_{n,b}(x) \equiv x(x + b) \mod n$, as defined in [10]

Only the public key n, b is required for encryption. If we fix b to 0 then $\mathrm{E}_{n,b}(x)$ becomes a simple squaring operation $\mathrm{E}_n(x) = x^2 \mod n$. Rabin's Scheme requires only one squaring, whereas RSA requires several squarings and multiplications for encryption. Therefore encryption with Rabin's Scheme is several hundreds of times faster than RSA [11].

Decryption involves finding the four square roots x_1, x_2, x_3, and x_4 of $c = \mathrm{E}_n(x) \equiv x^2 \mod n$. Certain simplifications are possible if $p \equiv q \equiv 3 \mod 4$. We would like to point the interested reader to [15] for a complete description of these algorithms. A hardware implementation of the decryption function is certainly feasible but beyond the scope of this paper.

2.3 The NtruEncrypt Public Key Cryptosystem

NtruEncrypt is a relatively new cryptosystem that claims to be highly efficient and particularly suitable for embedded applications such as smart cards or RFID tags, while providing a level of security comparable to that of other established schemes, in particular RSA. While it has not yet received the same level of scrutiny for establishing its resistance to cryptanalysis, there is evidence for efficiency in the simplicity of its underlying arithmetic. In this section we briefly describe the basic setup of NtruEncrypt and its operations. For more in-depth descriptions of the mathematical properties of NtruEncrypt we refer to [12, 16].

NtruEncrypt is based on arithmetic in a polynomial ring $R = \mathbb{Z}(x)/((x^N - 1), q)$ set up by the parameter set (N, p, q) with the following properties:

- All elements of the ring are polynomials of degree at most $N - 1$, where N is prime.
- Polynomial coefficients are reduced either mod p or mod q, where p and q are relatively prime integers or polynomials.
- p is considerably smaller than q, which lies between $N/2$ and N.
- All polynomials are univariate over the variable x.

Multiplication in the ring R is sometimes referred to as "Star Multiplication" based on use of an asterisk $\circledast$ as the operator symbol. It can be best described

as the discrete convolution product of two vectors, where the coefficients of the polynomials form vectors in the following way:

$$\begin{aligned} a(x) &= a_0 + a_1 x + a_2 x^2 + \ldots + a_{N-1} x^{N-1} \\ &= (a_0, a_1, a_2, \ldots, a_{N-1}) \end{aligned}$$

Then the coefficients c_k of $c(x) = a(x) \circledast b(x) \bmod q, p$ are each computed as the summation of partial products $a_i b_j$ with $i + j \equiv k \bmod N$. The modulus for reduction of each coefficient c_k of the resulting polynomial is either q for Key Generation and Encryption, or p for Decryption, as briefly described below. A thorough description of these procedures along with an initial security analysis can be found in [12].

Key Generation The following steps generate the *private key* $f(x)$:

1. Choose a random polynomial $F(x)$ from the ring R. $F(x)$ should have small coefficients, i.e. either binary from the set $\{0, 1\}$ (if $p = 2$) or ternary from $\{-1, 0, 1\}$ (if $p = 3$ or $p = x + 2$ [16, 17]).
2. Let $f(x) = 1 + pF(x)$ [1].

The *public key* $h(x)$ is derived from $f(x)$ in the following way:

1. As before, choose a random polynomial $g(x)$ from R.
2. Compute the inverse $f^{-1}(x) \pmod q$.
3. Compute the public key as $h(x) = g(x) \circledast f^{-1}(x) \pmod q$.

Encryption

1. Encode the plaintext message into a polynomial $m(x)$ with coefficients from either $\{0, 1\}$ or $\{-1, 0, 1\}$.
2. Choose a random polynomial $\phi(x)$ from R as above.
3. Compute the ciphertext polynomial $c(x) = p\phi(x) \circledast h(x) + m(x) \pmod q$.

Decryption

1. Use the private key $f(x)$ to compute the message polynomial $m'(x) = c(x) \circledast f(x) \pmod p$.
2. Map the coefficients of the message polynomial to plaintext bits.

3 Low-Power Design

This section provides a brief introduction to Low-Power Design. The power dissipation in CMOS devices can be summarized by the following equation [18]:

$$P = \underbrace{\left(\frac{1}{2} \cdot C \cdot V_{\mathrm{dd}}^2 + Q_{\mathrm{sc}} \cdot V_{\mathrm{dd}}\right) \cdot f \cdot N}_{P_{\mathrm{dyn}}} + \underbrace{I_{\mathrm{leak}} \cdot V_{\mathrm{dd}}}_{P_{\mathrm{leak}}} \tag{1}$$

[1] It is not strictly necessary to construct $f(x)$ in this way, but it is recommended in order to decrease the decryption failure rate. It is important, however, that $f(x)$ be invertible $(\mathrm{mod} p)$ and $(\mathrm{mod} q)$.

The term P_{dyn} represents the dynamic power dissipated during circuit activity. Circuit capacitance C, short-circuit charge Q_{sc} and supply voltage V_{dd} are technology dependent parameters [18] outside of our influence. The switching activity N and operating frequency f, however, can be influenced, and thus minimized, by architectural decisions. The second term P_{leak} represents the static power dissipation due to the leakage current I_{leak}. The leakage current is directly determined by the number of gates and the process technology. For more information about low-power design see [19]. Since we are using a standard cell based design flow, transistor level circuit optimizations are outside of the scope of this paper. In order to minimize the power consumption, we optimized our gate level design according to the following rules:

- The number of transitions ('0' to '1' and '1' to '0') has to be minimal.
- The circuit size should be minimized.
- Glitches cause unnecessary transitions and therefore should be avoided.

Our work is focused on the architectural aspects of low-power design, not on any specific VLSI techniques. Our architectures are implemented using a common CMOS standard cell design flow: circuit specification in structural VHDL, functional RT level simulation (ModelSim), synthesis (DesignCompiler Ultra, TSMC 0.13μm standard cell library), power optimization using annotated switching activity and delay information (DesignCompiler, PowerCompiler and ModelSim), and power analysis (PrimePower, back-annotated wire capacitances).

The TSMC library we use is fully characterized for timing and power consumption and includes several different wireload models for worst case estimation of interconnect capacitances. We would like to stress at this point that, although we use a low-voltage library, it is not in any way optimized for low-power designs.

4 Implementation

In order to provide a common ground for both implementations we had to make certain assumptions about the application scenario, which we state in the following paragraphs. Subsequently we describe the specifics of both implementations.

4.1 Assumptions

Sensor networks typically consist of a number of tiny nodes communicating with a base station [1]. The base station collects the data from the sensors and communicates with the outside world. The sensor nodes have only limited power and can therefore only communicate directly with nodes in close vicinity. They establish a routing tree with the base station at its root. The base station is assumed to have sufficient power for all computations and communications with the nodes and the outside world. Based on this setting we made the following assumptions:

– As stated in the introduction, we only consider the encryption operation of both systems. The purpose of this paper is to show that public key cryptography is computationally feasible in this environment.
– Depending on the exact application scenario it might be possible to fix the public key to a constant value. This is extremely beneficial for ultra-low power implementation, since the key can be embedded statically and does not require costly storage elements. In our implementations the public key is either hardwired or realized as a look-up table in combinational logic.
– Power consumption and energy efficiency are two different things. Depending on the actual application scenario one might want to trade off the two metrics differently over one another.

4.2 Rabin's Scheme

We have shown in Section 2.2 that the basic function for encryption in Rabin's Scheme is a simple squaring operation $E_n(x) = x^2 \mod n$, if we set $b = 0$. Squarers are a special form of multiplier. While any multiplier can be used to compute the square of a number, special-purpose squarers usually require significantly less hardware and are faster [20] by exploiting the symmetry of the squaring operation.

Squarers can be implemented in many ways. As our main concern is to conserve power we chose a bit-serial approach. The main advantage of a bit-serial design is that it minimizes the number of gates and reduces wire lengths—all factors that are of concern with regards to the circuit's power consumption. The bit-serial approach is ideal for modular reduction. Using the most significant bit (MSB) first method, modular reduction can be performed elegantly after each partial product addition. The generation of the partial product sequence, however, requires an extra 512-bit register. This is very expensive in terms of area and leakage power as each flip-flop is the equivalent of 6 gates. Therefore, we implemented the squarer as a bit serial modular multiplier where multiplicand and multiplier are hard-wired to the same input. All 512 bits of input are available in parallel at the same time. As a multiplier does not take advantage of the symmetry in squaring we expect it to consume more switching power. However, due to its smaller footprint the leakage power is also greatly reduced. At the low clock frequencies commonly encountered in sensor nodes, the influence of leakage power is the dominant part. An additional advantage of this approach is that this unit can easily be converted to a full multiplier for an implementation of RSA or a similar algorithm.

Figure 2 shows the architecture of our squarer. It is a standard bit serial multiplier design comprised of a Left Shift Register, a Bit Multiplier, a Left Shift unit, and the main units Adder and Sum Register. In order to perform modular multiplication we added two multiplexers which toggle the input of the adder between the next partial product and the 2's complement of the modulus n (reduction). The control logic determines whether a reduction operation is necessary after an addition. Since we are using the same adder for both functions, the number of clock cycles needed for one squaring is data dependent and at most 1024.

The most complex part of the squarer is the Adder. There are two basic adder designs that are suitable for a low power implementation, namely carry-save adder and ripple-carry adder. A ripple-carry adder uses fewer gates and hence consumes the least amount of leakage power, but as the worst case carry chain is the longest, this adder also has the longest delay. The propagation of carries causes glitches which in turn cause a very high dynamic power consumption. A carry-save adder on the other hand propagates carries only by one position, hence there are no glitches, resulting in insignificant amounts of delay and dynamic power consumption. Its disadvantage is that the result is kept in redundant carry-save representation which requires 512 additional flip-flops. This in turn causes a higher consumption of leakage power. Since partial products and complements of the modulus can be accumulated in redundant form, the final non-redundant result needs to be computed only at the very end of the multiplication which takes 512 additional clock cycles.

Neither of both approaches seems optimal for this implementation, so we tried to strike a balance between power and speed. For our adder we are using a ripple-carry adder and insert a carry-save bit on every 8th bit position. Hence the carries ripple for a maximum of 8 bits causing some glitching but significantly less than a full ripple-carry adder would. The dynamic power consumption is therefore much lower than for a full ripple-carry adder. This adder also needs only 64 additional flip-flops to store the carry bits, which is 448 flip-flops less than necessary for a full carry-save adder. This approach, however, introduces a new difficulty. After adding a partial product to the sum, the result has to be shifted. This would misalign the saved carry bits [2]. Hence, carry bits need to be re-aligned before shifting the sum. This is done by adding the carry bits to the sum in the appropriate position and saving the carry bits at the new position. The cost for this is a 512 bit multiplexer, 512 additional clock cycles and a slightly more complex control logic.

The Control logic is comprised of two state machines and one counter. The counter is implemented as a Linear Feedback Shift Register (LFSR) and "counts" up to 512. LFSRs have reduced switching activity and are faster than regular counters, hence reducing the effects on the critical path delay. Furthermore it is clock gated and can be reset. The counter is used to count all the multiplication steps and also to count the worst case number of steps necessary to ripple all 64 carry-save flip-flops. The main state machine of this control logic keeps track of the overall operation of the circuit. The second state machine takes care of arithmetic operations of the circuit. Furthermore it is responsible for the clock gating of the counter and the left shift register (see Figure 2).

4.3 NtruEncrypt

The basis for our ultra-low power NtruEncrypt architecture is the multiplication operation in the ring R, a cyclic convolution of two polynomials of the same

[2] This problem does not occur when a full carry-save adder is being used as there is one carry bit associated with every bit position

degree N. Considering a scenario, in which a sensor node encrypts a message and sends it to the base station, allows us to make the following observations which are helpful in creating an ultra-low power architecture. Similar observations can also be made for the case of decryption, but these are omitted here due to space restrictions.

- As mentioned at the beginning of this section, the public key of the node $h(x)$ is constant and embedded in the device. Since p is also constant, we can store a pre-scaled version of the public key $h'(x) = ph(x) \bmod q$. Thus we only need to compute $c(x) = \phi(x) \circledast h'(x) + m(x) \bmod q$.
- Coefficients of the public key $h(x)$ are computed modulo q and therefore occupy the larger of two wordsizes, while those of the random polynomial $\phi(x)$ are reduced modulo p. For our choice of $p = 3$ each coefficients of $\phi(x)$ is encoded as two bits. Since the public key is constant and realized as a look-up table, only $2N$ bits of storage are required as opposed to $N\lceil \log_2 q \rceil$.
- We assume that we have a good source of random bits available for generation of the random polynomial $\phi(x)$. In this paper we focus on the computational aspects of NtruEncrypt only, and therefore random number generation falls outside of the scope of this paper. For information on a compact implementation of an RNG based on digital artefacts requiring only a few hundred gates we refer to [21].

The algorithm consists of two nested loops: The outer loop iterates over all N coefficients of the result. The inner loop computes the coefficient by accumulating products of the form $a_i b_j$, with index i increasing and j decreasing mod$N$. The three major building blocks comprising the data path of the circuit—public key LUT, arithmetic units and circular buffer—are illustrated in Figure 3. The public key look-up table is realized in combinational logic that lends itself to optimization through the synthesis tool. The circular buffer consists of $2N$ bits of storage elements containing the coefficients of the random polynomial $\phi(x)$. Data enters the buffer through a multiplexer which connects the two ends of the buffer and forms a ring. Both, public key LUT and circular buffer, feed into the arithmetic units (AUs) which multiply and accumulate the operands. The smallest version of the circuit implements only a single AU. Yet, the architecture allows the implementor to scale up the number k of parallel AUs relatively easily, with minimal impact on the other elements of the design. Section 5.4 elaborates further on NtruEncrypt's inherent scalability. An AU consists of a partial product generator, a carry-save adder and a register. For any long operand a and short operand b the partial product generator will compute $ab \bmod q$. By choosing $p = 3$ and $q = 128$ the modular reduction of the intermediate result $c = \sum a_i b_j \bmod q$ comes essentially for free through simple truncation of bits at positions $\geq \log_2 q = 7$.

The control logic is designed to be as simple as possible in order to avoid being the bottleneck in terms of power consumption. The two nested counters needed for keeping track of coefficients in the inner and outer loop of the algorithm are implemented as Linear Feedback Shift Registers (LFSR) for reduced switching

activity. Furthermore, clock gating is used extensively whenever possible, to avoid any unnecessary switching activity and reduce parasitic wire capacitance.

In the case of only a single AU each round of computation takes $N + 8$ clock cycles to complete, with one coefficient per round. The eight additional clock cycles are necessary for addition of the message coefficient and propagation of carries in the carry-save adder. The total number of clock cycles for a full polynomial multiplication of N coefficients therefore takes $29,225$ clock cycles ($N = 167$). If k AUs are computing coefficients in parallel, the rounds overlap partially and the number of clock cycles amounts to $(N+8)(\lceil N/k \rceil)+k-1$. For a high degree of parallelization k the number of clock cycles can thus be reduced dramatically, i.e. to only 433 cycles for $k = 84$.

5 Analysis

In this section we analyze the proposed architectures according to various metrics of interest to ultra low-power applications such as sensor nodes. Since both architectures and algorithms are distinctly different from each other a direct comparison is difficult. We alleviate this situation by fixing system parameters to values that match security levels of both systems as closely as possible, as mentioned in Section 2.1.

5.1 Definition of Metrics

Chip Area The number of equivalent gates (2-input NAND gate) used by the circuit. This metric is independent of the process technology, and correlates well with the actual area of the physical layout.

Power Consumption This is the total power consumption of the circuit, categorized into static and dynamic power. This metric is highly dependent on the process technology. In this context, however, both architectures use the same target library so that differences in power consumption are a direct consequence of differences in the architecture.

Throughput Specifies the number of plaintext bits that are encrypted per second. This metric is independent of any message expansion properties of a given system.

Energy per Bit Encrypted Amount of energy necessary to encrypt a single bit of the message. This metric can be used to compare the energy efficiency of cryptosystems with a roughly equivalent level of security. It is independent of the actual operand length.

Scalability refers to the possibility of scaling an algorithm between bit serial and highly parallelized realizations in an efficient manner. A closely related concept is modularity, which is an indicator of how easily simple processing elements can be replicated for a higher degree of parallelization of a task in performance critical settings.

5.2 Rabin's Scheme

The main concern driving our low-power implementation of Rabin's Scheme is its storage requirement. Many well known techniques for optimizing a modular squarer require either more circuitry or more storage elements. At our targeted clock frequency of 500 kHz the static power consumption is dominant and therefore has to be minimized. Hence, we built a squarer as a bit-serial multiplier, operating on the entire width of the 512 bit multiplicand and on a single bit of the multiplier at a time. In order to conserve area we use the same adder for accumulating the partial products, modulo reducing the results, and re-aligning the carry bits before each shift. This approach consumes a chip area of less than $17,000$ gates with its accompanying static power consumption of $117.5\mu W$. The dynamic power consumption at 500 kHz is $30.68\mu W$ resulting in a total average power consumption of $148.18\mu W$ (Table 1). It increases linearly with the operating frequency as shown in Figure 1. A breakdown of the power consumption by functional blocks reveals that the adder consumes 40% of the power and all storage elements combined consume 38%. The power consumption of the complex control logic for this circuit is negligible at 2%.

5.3 NtruEncrypt

The hardware friendly arithmetic underlying the NTRU system lends itself very well to highly scalable and low-power implementations, since the computation of each individual coefficient is independent from one another. At the same time we can reorder the computation in a way that facilitates parallel computation of multiple coefficients. This can be achieved by simply replicating arithmetic units and slightly adjusting the control logic. The circular buffer allows parallel access to multiple coefficients in sequential order as illustrated in Figure 3, Section 4, thereby avoiding memory access bottlenecks.

A breakdown of the power consumption by functional blocks reveals that the most significant contribution is made by the circular buffer (77%), while an arithmetic unit contributes the least amount (6%). The cost for an implementation with only a single arithmetic unit is therefore relatively high compared to a more parallelized variant. The small cost of adding arithmetic units, on the other hand, allows for a high degree of parallelization. This level of scalability is advantageous when it comes to achieving optimal energy efficiency. In the following analysis we therefore also consider performance estimates for a highly parallelized ($k = 84$) variant of our NtruEncrypt architecture, based on data obtained from simulation of the digit serial implementation ($k = 1$).

Our smallest implementation of NtruEncrypt with a single arithmetic unit takes up a chip area of less than 3000 equivalent gates, including the circular buffer and the combinational look-up table of the public key. Gate level power simulation indicates an average power consumption of less than $20\mu W$ at a clock frequency of $500kHz$, close to the amount of static leakage power (see Table 1). As before with Rabin's Scheme we see dynamic power consumption beginning to dominate at faster clock speeds as it increases linearly with the frequency (Fig. 1). This is in agreement with our expectation from (1).

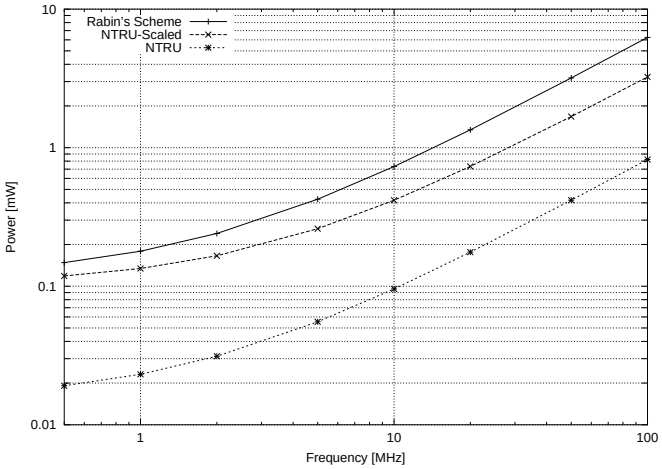

Fig. 1. Power Consumption over a Range of Clock Frequencies

5.4 Comparison

Table 1 shows a direct comparison between Rabin's Scheme and both variants of NtruEncrypt in the metrics defined above. The architectures of Rabin's Scheme and the simple variant of NtruEncrypt were both intended to achieve the least possible power consumption given the available standard cell library, without necessarily reaching optimal energy efficiency. The results are summarized in the first two columns. After an initial analysis of the architectural differences we decided to include estimates for a highly parallelized variant of NtruEncrypt in the third column of the comparison. The degree of parallelization $k = 84$ was chosen in a way that the area footprint roughly matches that of Rabin's Scheme, and secondly that $\lceil N/k \rceil - N/k$ is as small as possible. This is to divide the number of coefficients N in a way that utilization of the AUs is high during the last round of computation.

Rabin's Scheme takes up almost six times the area of simple NtruEncrypt with a single AU. On the other hand it also has the advantage of performing almost forty times better. This is to be expected due to its large operands and full-word arithmetic. If, however, the absolute area and power requirements are the limiting factor, it might not be flexible enough. Also, our estimates for the parallelized variant of NtruEncrypt indicate that it outperforms Rabin's Scheme by nearly factor two using the same area footprint. From the figures in Table 1 it is evident that static leakage power is the main culprit for the relatively high energy consumption of both implementations. We would like to stress the fact that leakage power is highly technology dependent and that the ASIC standard cell library we use is not optimized for low power design. The dynamic power consumption of an architecture, on the other hand, is proportional to its switching activity. It therefore makes sense to differentiate between these two

Table 1. Summary of comparison between Rabin's Scheme and NtruEncrypt

	Rabin	Ntru ($k = 1$)	Ntru ($k = 84$)
Equivalent security	60 *bits*	57 *bits*	57 *bits*
Area [eqv. gates]	16,726	2,850	16,200
- combinational	8,875	523	7,000
- storage elements	7,851	2,327	9,200
Delay (avg. # cycles)	1,440	29,225	433
Avg. power @ 500kHz	148.18 μW	19.13 μW	118.7 μW
- static (%)	117.5 μW (79.3%)	15.10 μW (78.9%)	103.06 μW (86.8%)
- dynamic (%)	30.68 μW (20.7%)	4.03 μW (21.1%)	15.64 μW (13.2%)
- peak power	169.8 μW	20.22 μW	n/a
Energy	426.76 nJ	1,118.15 nJ	102.79 nJ
- per bit encrypted	833.5 pJ (512 bits)	4,235.41 pJ (264 bits)	389.4 pJ (264 bits)
Throughput	177.8 $kbits/s$	4.52 $kbits/s$	304.85 $kbits/s$

influences if we want to compare the relative merits of one architecture over the other, independently of the process technology. It turns out that dynamic power consumption in Rabin's Scheme is nearly twice as high as in NtruEncrypt's case, despite the same area and the fact that leakage power differs by only 12%.

The throughput that either architecture can achieve at a given clock frequency depends on the number of clock cycles for an encryption and the number of plaintext bits per block. In Rabin's Scheme the plaintext is up to 512 bits long. At a clock frequency of 500 kHz and an average of 1440 cycles per operation this translates into a maximum theoretical throughput of 177.8 kbits/s. Since NtruEncrypt uses N ternary coefficients we can determine its throughput in terms of kbits/s by first converting the capacity of the message polynomial $m(x)$ into bits. $N = 167$ ternary coefficients can hold information equivalent to $\lfloor N \log_2 3 \rfloor = 264$ bits. The entire encryption operation takes 29225 clock cycles for NtruEncrypt with a single AU, and 443 cycles with 84 AUs. Operating at the same clock frequency, the simple variant compares unfavorably to Rabin's Scheme at only 4.52 kbits/s throughput, almost 40 times less. The estimates for the highly parallelized variant, however, indicate a performance level of 304.85 kbits/s, nearly twice the throughput of Rabin's Scheme.

For any cryptographic scheme there is a multitude of possible design choices by which power consumption can be traded off against performance and vice versa. Ultimately, however, we would like to know the amount of energy that is necessary for an elementary encryption operation, i.e. the cost of encrypting a bit of data at a certain level of security. The amount of energy for the entire operation is the product of average power consumption and the time it takes to complete that operation. Considering the amount of plaintext data that can be encrypted in one operation, we determine the amount of energy per bit encrypted as

$$E_{\text{bit}} = \frac{P_{\text{avg}} \cdot n_{\text{cycles}}}{f_{\text{clock}} \cdot l_{\text{op}}}$$

where l_{op} is the operand length in bits, i.e. 512 for Rabin's Scheme and 264 for NtruEncrypt. As we have discussed earlier, Rabin's Scheme uses more power than NtruEncrypt with a single AU, but it also takes much fewer clock cycles to complete. We can make a similar observation by looking at the energy per bit metric. The amount of energy necessary to encrypt a single bit with NtruEncrypt is about five times higher than with Rabin's Scheme. The picture changes yet again when we consider NtruEncrypt's parallelized variant. Our estimates suggest that the amount of energy per bit drops by nearly factor 11 and is thus less than half the amount of Rabin's Scheme.

To put our results into perspective, we compare them to estimates reported in [8] that were obtained from simulation of various public key algorithms on existing general purpose processor architectures. An implementation of the emerging scheme XTR on the ARC3 processor suggests an energy consumption of around $130\,\mu$J at a security level that is comparable to RSA-1024 or 72 bits of equivalent security. Despite the difference in security levels, this is still between factor 100 and 1000 more energy than what our architectures require, proving the strength of customized application specific architectures.

6 Conclusions

We have demonstrated in this paper that it is possible to design public key encryption architectures with power consumption of less than $20\,\mu$W using the right selection of algorithms and associated parameters, optimization and low-power techniques. In spite of the common perception of public key cryptography, it is possible to achieve a level of power consumption low enough to allow its use even in self-powered sensor nodes. Our implementation is based on a regular ASIC standard cell library that is not specifically optimized for low-power. It is thus possible to achieve even better results than ours, although that is not the point we are trying to make here. The use of public key schemes facilitates much simpler security protocols than those currently in use with the sensor network community, and has a potential impact on a much wider range of applications. RFIDs and contactless smart cards are further examples of ubiquitous computing applications requiring energy efficient cryptographic functions. So far public key cryptography has not even been considered for these devices due to its perceived complexity.

Our findings show further that a traditional modular arithmetic based scheme, such as Rabin's, does have a significant disadvantage compared to new and emerging schemes represented here by NtruEncrypt. The use of arithmetic in a polynomial ring allows for a very compact, yet scalable implementation in hardware. Additionally, NtruEncrypt's decryption operation—although not further considered in this paper—is based on the same arithmetic operation. This opens up the possibility for realization of two way key exchange protocols, while this is more difficult with Rabin's Scheme, due to its asymmetric properties of encryption and decryption.

Further research into energy efficient cryptographic primitives is necessary, but our findings give us the confidence that public key cryptography in ubiquitous computing applications is possible and that it can be done efficiently using customized hardware architectures.

References

1. Perrig, A., Szewczyk, R., Tygar, J.D., Wen, V., Culler, D.E.: SPINS: security protocols for sensor networks. Wireless Networks **8** (2002) 521–534
2. Fulford, B.: Sensors gone wild. Forbes Global (2002) http://www.forbes.com/global/2002/1028/076_print.html.
3. Polastre, J.: Design and implementation of wireless sensor networks for habitat monitoring. Master's thesis, University of California at Berkeley (2003)
4. Mainwaring, A., Polastre, J., Szewczyk, R., Culler, D., Anderson, J.: Wireless sensor networks for habitat monitoring. In: First ACM Workshop on Wireless Sensor Networks and Applications, Atlanta, GA, USA. (2002)
5. Burne, R., et al.: Self-organizing cooperative sensor network for remote surveillance: improved target tracking results. In: Proceedings of the SPIE. Volume 4232., Boston, SPIE, SPIE-Int. Soc. Opt. Eng, USA (2001) 313–321
6. Meininger, S., Mur-Miranda, J., Amirtharajah, R., Chandrakasan, A., Lang, J.: Vibration-to-electric energy conversion. IEEE Transactions on Very Large Scale Integration (VLSI) Systems **9** (2001) 64–76
7. Amirtharajah, R., Chandrakasan, A.P.: Self-powered signal processing using vibration-based power generation. IEEE Journal of Solid-State Circuits **33** (1998) 687–695
8. Carman, D.W., Kruus, P.S., Matt, B.J.: Constraints and approaches for distributed sensor network security. Technical report, NAI Labs, Security Research Division, Glenwood, MD (2000)
9. Weimerskirch, A., Paar, C., Shantz, S.C.: Elliptic curve cryptography on a Palm OS device. In V. Varadharajan, Y.M., ed.: The 6th Australasian Conference on Information Security and Privacy (ACISP 2001). Volume 2119 of LNCS., Heidelberg, Springer-Verlag (2001) 502–513
10. Rabin, M.O.: Digitalized signatures and public key functions as intractable as factorization. Mit/lcs/tr-212, Massachusetts Institute of Technology (1979)
11. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM **21** (1978) 120–126
12. Hoffstein, J., Pipher, J., Silverman, J.: NTRU: A ring-based public key cryptosystem. In Buhler, J., ed.: Algorithmic Number Theory (ANTS III). Volume 1423 of LNCS., Berlin, Springer-Verlag (1998) 267–288
13. Lenstra, A.K., Verheul, E.R.: Selecting cryptographic key sizes. Journal of Cryptology: The Journal of the International Association for Cryptologic Research **14** (2001) 255–293
14. Hoffstein, J., Silverman, J., Whyte, W.: NTRU report 012, version 2. estimated breaking times for NTRU lattices. Technical Report 12, NTRU Cryptosystems, Inc., Burlington, MA, USA (2003)
15. Menezes, A.J., van Oorschot, P.C., Vanstone, S.: Handbook of Applied Cryptography. CRC Press Inc. (1997)
16. Hoffstein, J., Silverman, J.H.: Optimizations for NTRU. In: Proceedings of Public Key Cryptography and Computational Number Theory, de Gruyter, Warsaw (2000)

17. Bailey, D., Coffin, D., Elbirt, A., Silverman, J., A.Woodbury: NTRU in constrained devices. In Ç. Koç, Naccache, D., Paar, C., eds.: CHES 2001. Volume 2162 of LNCS., Berlin, Springer-Verlag (2001) 266–277
18. Devadas, S., Malik, S.: A survey of optimization techniques targeting low power VLSI circuits. In: Proceedings of the 32nd ACM/IEEE Conference on Design Automation. (1995) 242–247
19. Rabaey, J., Pedram, M.: Low Power Design Methodologies. Kluwer Academic Publishers, Norwell, Massachusetts (1996)
20. Parhami, B.: Computer Arithmetic: Algorithms and Hardware Designs. Oxford University Press (2000)
21. Epstein, M., Hars, L., Krasinski, R., Rosner, M., Zheng, H.: Design and implementation of a true random number generator based on digital circuit artifacts. In Walter, C.D., Çetin K. Koç, Paar, C., eds.: CHES 2003. Volume 2779 of LNCS., Berlin, Springer-Verlag (2003) 152–165

Appendix

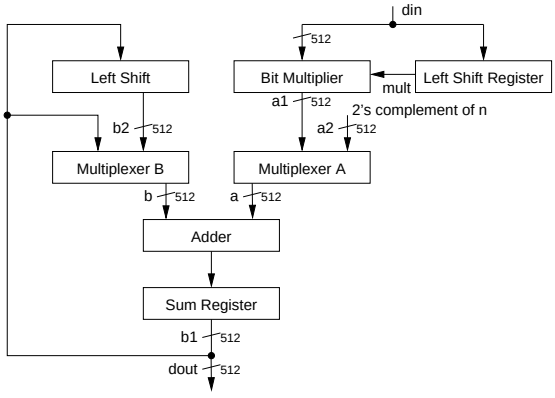

Fig. 2. Block Diagram for Rabin's Scheme

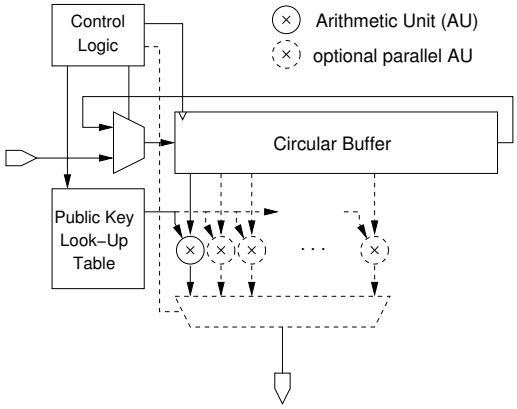

Fig. 3. Block Diagram for NtruEncrypt