Towards Lightweight Identity-Based Encryption for the Post-Quantum-Secure Internet of Things

Tim Güneysu
University of Bremen and DFKI, Germany
tim.gueneysu@uni-bremen.de

Tobias Oder
Ruhr-Universität Bochum, Germany
tobias.oder@rub.de

***Abstract*—Identity-Based Encryption (IBE) was introduced as an elegant concept for secure data exchange due to its simplified key management by specifically addressing the asymmetric key distribution problems in multi-user scenarios. In the context of ad-hoc network connections that are of particular importance in the emerging Internet of Things, the simple key discovery procedures as provided by IBE are very beneficial in many situations. In this work we demonstrate for the first time that IBE has become practical even for a range of embedded devices that are populated with low-cost ARM Cortex-M microcontrollers or reconfigurable hardware components. More precisely, we adopt the IBE scheme proposed by Ducas et al. at ASIACRYPT 2014 based on the RLWE problem for which we provide implementation results for two security levels on the aforementioned embedded platforms. We give evidence that the implementations of the basic scheme are efficient, as for a security level of 80 bits it requires 103 ms and 36 ms for encryption and decryption, respectively, on the smallest ARM Cortex-M0 microcontroller.**

I. Introduction

With the advent of a Smart World and the Internet of Things (IoT), billions of devices are becoming connected, exchanging massive amounts of data, often combined with high demand on data security. While encryption schemes and digital signatures are established concepts to achieve the required data confidentiality and authentication, some very basic requirements remain a major challenge for the myriad of devices – namely the authenticated key management. Several solutions to this, for example such as Public-Key Infrastructures (PKI), still they remain complex due to the need of broadly issuing and distributing certificates signed by a trusted authority.

In this regard, the concept of Identity-Based Encryption (IBE) schemes can provide an elegant alternative to this problem by allowing parties and devices to use a personal identifier such as a user name or a serial number as public key. IBE schemes still require a trusted authority that generates and establishes the corresponding secret keys to given public keys that are either installed once during system setup or even on-demand basis. Combining the on-demand provision of secret keys with further attributes, this can even be used as mechanism for fine-grained access control. In this sense, computationally weak (sensor) nodes in the IoT with very limited security and protection features can simply send confidential messages to more powerful aggregator, encrypted under the aggregator node's unit number that can be easily retrieved by anyone using a simple network discovery procedure. Besides solving the key distribution problem, this has further advantages over conventional cryptography using symmetric master keys – in particular if the security of the aggregator node is compromised what directly results in a full break of the entire system.

IBE has a lengthy history, having been proposed by Shamir in his seminal work from 1984 [1]. Since then IBE has seen several interesting proposals for a potential practical instantiation including associated proofs of the security, such as the Boneh-Franklin and Boneh-Boyen schemes [2], [3] based on the Bilinear Diffie-Hellman Assumption using pairings, or Cocks's schemes using Quadratic Residues [4]. Still those schemes suffered from impractical parameters or large factors of ciphertext expansion that rendered them unusable for practical applications. Another proposal making use of the versatility of lattices was proposed by Ducas et al. in 2014 [5] that offered reasonable parameter sizes and moderate ciphertext expansion factors. Still it is not clear if the latter scheme is applicable to embedded systems to enable the aforementioned extremely desirable use-case on future devices in the IoT.

Contribution. In this work we like to close the gap considering the practical use of IBE and provide evidence that its basic instance is ready to be implemented on a wide range of embedded platforms. In particular, we explore how different security levels and parameters for the underlying RLWE assumption will affect the implementation of IBE encryption and decryption on (a) an ARM Cortex-M0 as extremely low-cost device, (b) an ARM Cortex-M4 and (c) on reconfigurable hardware (Xilinx Spartan-6). Using the parameter set with at least 80 bits of equivalent security for our IBE implementation, it takes 103 ms for an encryption operation on the ARM Cortex-M0 at 32 MHz, for the security level larger than 192 bits 164 μs are required on the Spartan-6 FPGA, clocked at 174 MHz. Furthermore, we propose parameters for both security levels that render an efficient IBE implementation via the NTT possible.

Related Work. To the best of our knowledge this is the first time that IBE is implemented for embedded devices. To date, there are only several previous works that proposed implementations of the basic RLWE assumption for microcontrollers and reconfigurable hardware as discussed below. The general concept of the RLWE scheme has been introduced in [6], [7]. FPGA implementations have been provided by Göttert et al. [8] and Pöppelmann and Güneysu [9]. In [10] Roy et al. presented the currently fastest implementation by using techniques to accelerate NTT-based polynomial multiplication. Finally, RLWE has been implemented for 8-bit AVR microcontrollers in [11], [12].

II. Background

A. Notation

First we introduce the notation that is used throughout the paper. The implemented IBE scheme operates on the ring $\mathcal{R} = \mathbb{Z}_q[\mathbf{x}]/\langle x^n + 1\rangle$ where n is a power of two and q is prime. The operator $\times$ denotes polynomial multiplication in $\mathcal{R}$ and $||f||$ the Gram-Schmidt norm of f. The operator $\lfloor f \rceil$ is the coefficient-wise rounding of a polynomial f. $\mathcal{A}(f)$ denotes the anticirculant matrix of f as described more detailed in [5].

B. Ring-LWE Encryption

The IBE scheme of [5] is based on a ring-variant of the Learning with Errors problem (RLWE) [6], [13].In its Hermite normal form defined over ideal lattices in the ring $\mathcal{R} = \mathbb{Z}_q[\mathbf{x}]/\langle x^n + 1\rangle$, the problem requires one to decide whether the samples $(\mathbf{a}_1, \mathbf{t}_1), \ldots, (\mathbf{a}_m, \mathbf{t}_m) \in \mathcal{R} \times \mathcal{R}$ are chosen uniformly random or whether each $\mathbf{t}_i =$

$\mathbf{a}_i\mathbf{s} + \mathbf{e}_i$ where $\mathbf{s}, \mathbf{e}_1, \ldots, \mathbf{e}_m$ have small coefficients from the (one-dimensional) discrete Gaussian distribution $\mathcal{D}_\sigma$ [6]. The distribution $\mathcal{D}_\sigma$ is defined such that a value $x \in \mathbb{Z}$ is sampled from $\mathcal{D}_\sigma$ with the probability $\rho_\sigma(x)/\rho_\sigma(\mathbb{Z})$ where $\rho_\sigma(x) = \exp(\frac{-x^2}{2\sigma^2})$ and $\rho_\sigma(\mathbb{Z}) = \sum_{k=-\infty}^{\infty} \rho_\sigma(k)$. Note that some authors do not define the discrete Gaussian by the standard deviation σ but instead use the parameter $s = \sqrt{2\pi}\sigma$.

A simple public key encryption scheme whose semantic security directly follows from the RLWE problem has been introduced in the full version [14] of Lyubashevsky et al. [6]. The required operations are Gaussian sampling and polynomial arithmetic, including addition and multiplication of polynomials with n coefficients where each coefficient is reduced modulo q and polynomials are reduced modulo $\mathbf{x}^n + 1$. For encryption, the n-bit binary message $m \in \{0,1\}^n$ has to be encoded into a polynomial $\bar{\mathbf{m}} \in R$. This is necessary as even after decryption small noise terms are present. By using threshold encryption and multiplying each one bit in the message by $\frac{q-1}{2}$ it is later on possible to recover from these errors.[1]

Note finally that a major advantage of lattice-based encryption is that no quantum algorithms are yet known to solve the underlying problems in polynomial time [15] – contrary to popular schemes like ECC or RSA [16].

C. Identity-Based Encryption

As introduced above, an Identity-based Encryption scheme provides the unique advantage that the sender of a message can simply use the identity of the receiver (such as an e-mail address, serial number, or MAC address) as public key for encryption [1]. This renders the need for a public key management and distribution infrastructure obsolete. To make this possible, IBE schemes require a trusted central authority that secretly stores a master secret key and the corresponding master public key that is known to all participating parties. The central authority derives each users private key from the master secret key and the ID of the user. Obviously, an IBE scheme therefore cannot provide non-repudiation as the central authority has access to all private keys. Note that the private keys still have to be installed or sent over a secure and authenticated channel to the device or user. In the IoT setting, this might be done by placing a secret key inside the device either straight after production or during installation of the system within a trusted environment.

IBE schemes are defined by the following four algorithms, MasterKeyGen, UserKeyGen, Encryption, and Decryption. MasterKeyGen is used to generate the key pair of the central authority, while UserKeyGen extracts the users private key from the master secret key and the ID. While the former algorithms are executed within the trusted environment of the central authority, Encryption and Decryption are the only algorithms that need to be performed by the device of the user. IBE schemes can rely on a range of intractable problems such as the Bilinear Diffie-Hellman Assumption [2], [3] or RLWE [5] for which typically security proofs can be given to reduce the security of IBE to those problems.

In this work, we investigate the feasibility of IBE on embedded devices in the Internet of Things and target the implementation of the IBE scheme proposed by [5]. Although we will briefly introduce the required technical content relevant for our work in the following, we refer to the original work for a more thorough discussion of the details and the security proof. For the generation of the master key by the central authority according to [5], the following MasterKeyGen procedure (Algorithm 1) is performed that defines the cryptographic instance for the entire IBE system.

[1]There is still a very small possibility for decryption errors. See [7], [8] for more details.

Algorithm 1 Master Key Generation

MasterKeyGen()
  do
    $f, g \leftarrow \mathcal{D}_{\sigma_f}$ ▷ $\sigma_f = 1.17\sqrt{\frac{q}{2n}}$
  while max $(||(g,-f)||, ||(\frac{q\bar{f}}{f\times\bar{f}+g\bar{g}}, \frac{q\bar{g}}{f\times\bar{f}+g\bar{g}})||) > 1.17\sqrt{q}$
  Using EEA to compute $\rho_f, \rho_g \in \mathcal{R}$ and $R_f, R_g \in \mathbb{Z}$ such that
  $\rho_f \cdot f = R_f \mod x^n + 1$ and
  $\rho_g \cdot g = R_g \mod x^n + 1$
  If $(GCD(R_f, R_g) \neq 1$ or $GCD(R_f, q) \neq 1)$ **Restart**
  Using EEA to compute $u, v \in \mathbb{Z}$ such that $u \cdot R_f + v \cdot R_g = 1$
  $F \leftarrow qv\rho_g$
  $G \leftarrow qu\rho_f$
  $k \leftarrow \lfloor \frac{F\times\bar{f}+g\times\bar{g}}{f\times\bar{f}+g\bar{g}} \rceil \in \mathcal{R}$
  $F \leftarrow F - k \times f$
  $G \leftarrow G - k \times g$
  $h \leftarrow g \times f^{-1} \mod q$
  $\mathbf{B} \leftarrow \begin{pmatrix} \mathcal{A}(g) & -\mathcal{A}(f) \\ \mathcal{A}(G) & -\mathcal{A}(F) \end{pmatrix}$
  Return $(sk, pk) = (\mathbf{B}, h)$
end

Algorithm 2 User Key generation. `GaussianSampler` as described in [5].

UserKeyGen($\mathbf{B}, ID$)
  $t \leftarrow H(ID) \in \mathbb{Z}_q^n$ ▷ $H()$ is a hash function.
  $(s_1, s_2) \leftarrow (t, 0) -$ `GaussianSampler`$(\mathbf{B}, \sigma, (t, 0))$
  Return $sk_{ID} = s_2$
end

For every user or device, the central authority issues the UserKeyGen procedure as shown in Algorithm 2 to generate a corresponding secret key. This step is typically done as part of a personalization phase once for each embedded devices inside a trusted environment when setting up the entire system.

In this work, we will focus on the following Encryption and Decryption procedures (Algorithm 3 and 4, respectively) and exclude the previously discussed MasterKeyGen and UserKeyGen from our implementation as those computations will not be performed on the computationally constrained embedded devices in the field. For Encryption and Decryption we need to consider three operations: sampling of polynomials with uniformly distributed ternary coefficients, polynomial multiplication and threshold encoding. Threshold encoding is defined to convert a message of n bits into a polynomial with n coefficients in $[0, 1]$. Then this polynomial is multiplied by $\frac{q-1}{2}$ to generate an encoded form of the polynomial. During decryption the encoded message polynomial is received which is augmented with errors and has to be checked for each coefficient whether it fits $[\frac{q}{4}, \frac{3q}{4})$. If so, the corresponding ciphertext bit is set to 1 and otherwise to 0.

D. Parameter Selection

Parameter selection for lattice-based problems is a non-trivial task due to the moving complexity bound for their best-known attacks and other side effects such as the decryption failure rate for RLWE. In [5] Ducas *et al.* outlined two parameter sets in their work, one for 80 bits of equivalent symmetric pre-quantum security and one claiming 192 bits of security – but without giving details. Based on their and

Algorithm 3 Encryption

1: Encryption(h, ID, m)
2: $r, e_1, e_2 \leftarrow -1, 0, 1^N$ (uniform)
3: $u \leftarrow r \times h + e_1 \in \mathcal{R}$
4: $v \leftarrow r \times H(ID) + e_2 + \texttt{encode}(m) \in \mathcal{R}$
5: **Return** $(c_1, c_2) = (u, v)$
6: **end**

Algorithm 4 Decryption

1: Decryption(sk_{ID}, c_1, c_2)
2: $w \leftarrow v - q \times sk_{ID}$
3: $m \leftarrow \texttt{decode}(w)$
4: **Return** m
5: **end**

previous assumptions as reported in [17], we picked the parameters as shown in Table I that provided a best fit to our implementation goals with respect to an efficient implementation on embedded systems.

We decided to set q to satisfy the requirement $q \bmod 2n = 1$ enabling a fast multiplication via NTT. We further optimized our choice regarding the binary representation of our modulus to match a linear combination of powers of two (as for Generalized Mersenne Primes [18]). This yields further benefits for fast modular reduction (see Section III).

III. Implementation

In this section, we present three implementations of IBE for typical IoT devices. This includes an implementation for (a) the low-cost ARM Cortex-M0 microcontroller, (b) the most powerful ARM Cortex-M4 microcontroller and (c) an FPGA implementation targeting a Xilinx Spartan-6 FPGA. We excluded the key setup and key extraction from our implementations (as these values will be computed and distributed by the master authority) and therefore reserved storage for the master public key and device ID (use for encryption) and the secret key of the receiver for decryption. For efficient implementation, all keys are stored straight in their NTT representation as all of them are used as input for NTT-based polynomial multiplication.

A. Microcontroller Implementation

We chose the ARM Cortex-M0 and Cortex-M4F as target platforms for our software implementation as those microcontrollers offer the computational capabilities of a 32-bit architecture combined with the advantages of low-cost and low-power platforms. The first target platform is the XMC 2Go evaluation board (Cortex-M0) that is designed for extremely lightweight applications. For applications that require computational power including a more sophisticated instruction set, we also implemented IBE on the STM32F4DISCOVERY board (Cortex-M4). Both implementations have been developed using Keil μVision V5.17 and armcc V5.06 as compiler. For both microcontrollers we used on-board debugging facilities for fast prototyping. The main differences concerning the instruction set of both microcontrollers is that the Cortex-M0 lacks or has only limited access to, for instance, advanced instructions like `Multiply-Accumulate`, conditional execution, and status register-preserving instructions.

TABLE I
Parameter selection for an efficient implementation of IBE.

Minimum Security level	80 bits	192 bits
Polynomial degree n	512	1024
Modulus q	16813057	134348801
q as sum of powers of two	$\{24, 15, 11, 10, 0\}$	$\{27, 17, 0\}$
n-th root of unity ω	2869	378500
$\psi = \sqrt{\omega}$	6129063	43530086

The implementation is split into main building blocks consisting of NTT, ternary sampling and point-wise multiplication. The cost for the encoding and decoding of the message are negligible. We applied a trick for the NTT as described in [12] to avoid expensive bit-reversal and additional multiplications with powers of ψ resp. ψ^{-1}. No temporary memory for polynomials is required as all computations can be performed on the memory in situ that is used for input and output. Therefore the dynamic memory consumption of our encryption and decryption routine is negligible.

During encryption, we need to sample polynomials with uniformly distributed coefficients in $\{-1, 0, 1\}$. This can be achieved by sampling two uniformly random bits. If the sampled value is $10_2 = 2_{10}$, the input is rejected and the procedure restarted with further two random bits. Note that since the final output does not depend on any rejected value, this rejection does not lead to a timing leakage that could be exploited by an attacker. If the value gets accepted, 11_2 is interpreted as -1 and the result is returned. The uniformly random bits are drawn from on-board TRNG (M4) resp. PRNG (M0).

Given a dedicated 32-bit multiplication instruction in the microcontroller's instruction set, the most crucial operation in the arithmetic remains the modular reduction. Since both target platforms feature single-cycle multiplier, other arithmetic can be carried out within very few cycles, even when multi-precision operands are required. Unfortunately, we are faced with a prime modulus of 25 bits to enable NTT, so that a vanilla-plain reduction would take several hundreds of clock cycles and thus need to be a target for optimization. We implemented a reduction that exploits the fact that our modulus is a composition of powers of two as known from generalized Mersenne primes (MersenneRed, Algoritm 5).

Recall that the modulus for the selected 80-bit security parameter is decomposed in powers of 2 by $q = 16813057 = 2^{24} + 2^{15} + 2^{11} + 2^{10} + 1 = 0 \bmod q$. By rearranging this equation, we get $2^{24} = -2^{15} - 2^{11} - 2^{10} - 1$ $(Eq.\ 1)$ we rewrite the 49-bit output of a multiplication as $x = a \cdot 2^{24} + b$ $(Eq.\ 2)$ where a represents the higher 25 bits of the result and b the lower 24 bits. If we combine equation (1) and (2) we get $x = b - a \cdot 2^{15} - a \cdot 2^{11} - a \cdot 2^{10} - a \bmod q$. With a high probability, the result will not be in $[0, q-1]$ but smaller than 0. One can apply the described procedure multiple times to incrementally reduce the absolute value. Depending on whether the reduction has been applied odd times or even times, we have to add or subtract q afterward. We investigated the iteration count how often we need to repeat the described approach before we apply simple additions/subtractions of q. It turned out that three iterations led to best performance. After that, on average 0.2 additions of q are necessary to get a result in $[0, q-1]$. Notice that a maximum of 42 additions are necessary. For a constant-time implementation we therefore recommend to run the reduction four times since then at most one additional subtraction is required which corresponds to a small performance penalty. As the shape of the prime in the 192-bit security parameter set is even simpler $(2^{27} + 2^{17} + 1)$ the reduction in this case requires even less operations.

Algorithm 5 Mersenne Reduction

Precondition: Assume k to be the number of reduction runs

MersenneRed(x)
  for $j = 1$ **to** k **do**
    $high \leftarrow x >> 24$
    $low \leftarrow x$ & 0xFFFFFF
    $x \leftarrow low - high$
    $x \leftarrow x - (high << 10)$
    $x \leftarrow x - (high << 11)$
    $x \leftarrow x - (high << 15)$
  end for
  while $x < 0$ **do** ▷ k odd
    $x \leftarrow x + q$
  end while
  while $x \geq q$ **do** ▷ k even
    $x \leftarrow x - q$
  end while
  return x
end

B. FPGA Implementation

We now detail the IBE implementation for Xilinx Spartan-6 FPGAs to demonstrate its feasibility even on low-cost hardware.

The core architecture of our FPGA implementation is shown in Figure 1. It adopts the basic design for a RLWE processor as proposed in [9] which uses a generic micro-code engine to implement lattice-based cryptosystems. Since the rationales for the hardware implementation of IBE are very similar, it makes perfect sense to follow the same design principles as in [9]. Note that more specific implementations ([19], [20]) with more parameter-specific optimizations, like dedicated modular reductions, typically deliver better performance results. However, since we are aiming for flexibility with respect to supporting various parameter sets, we opted to take the former design strategy based on a generic micro-code engine. With respect to the original design for RLWE, we still were required to include several modifications. At first, we added a ternary sampler as encryption and decryption in IBE does not require Gaussian but ternary sampling. Second, we updated the parameters and precomputed tables as described in Section II-D. Finally, we changed the micro-code to support IBE. Note that we could not use the optimized NTT operation as proposed in [12] as this would requires two different butterfly architectures. We opted to keep the resource consumption low, hence our core uses the same butterfly architecture for the forward and backward transform.

IV. Results and Comparison

A. Microcontroller Results

We start with the performance of our implementation on the ARM Cortex-M0 and Cortex-M4. To measure the performance of the Cortex-M0 implementation, we utilize the internal `systick` timer that periodically generates interrupts. We count these interrupts to determine the running time of our implementation. Since the handling of the interrupts causes an additional overhead that distorts the results, we limit the interrupt interval to 1000 clock cycles even though we sacrifice some accuracy on the cycle counts by doing so. The Cortex-M4 offers a built-in cycle counter `DWT_CYCCNT` that we can easily check before and after each operation to measure its execution time. The performance of each operation is computed by averaging the cycle counts over 1000 runs with random input.

The performance results for our implementations are given in Table II. Translating the cycle counts to milliseconds, one encryption takes 103 milliseconds on the Cortex-M0 and 5.80 milliseconds on the Cortex-M4, consisting of three number-theoretic transforms (M0: 85%, M4: 79%), three runs of the ternary sampler (M0: 4%, M4: 11%), and operations with lower cycle count, like encoding, decoding, and point-wise multiplication. As decryption requires only one NTT instead of three, it takes 36 milliseconds on the Cortex-M0 and 1.90 milliseconds on the Cortex-M4. It comes to no surprise that the cycle counts of the Cortex-M0 implementation are roughly 3.5 times higher than the cycle counts of the Cortex-M4 implementation as the M4 features a more powerful instruction set. The sampling of a ternary error polynomial takes comparatively long on the Cortex-M4. This is due to the fact that we use true-random numbers on the Cortex-M4 but only pseudo-random numbers on the Cortex-M0.

TABLE II
Cycle counts and ROM consumption for both microcontroller implementations and parameter sets. The Cortex-M0 runs at 32 MHz and the Cortex-M4 runs at 168 MHz.

	(512/16813057)		(1024/134348801)	
Operation/cycles	Cortex-M0	Cortex-M4	Cortex-M0	Cortex-M4
Encryption	3,297,380	972,744	6,202,910	1,719,444
Decryption	1,155,000	318,539	2,171,000	557,015
Sampling	41,960	34,655	83,390	69,320
$\text{NTT}^{CT,\psi}_{no\rightarrow bo}$	923,300	251,763	1,749,860	444,117
$\text{INTT}^{GS,\psi^{-1}}_{bo\rightarrow no}$	948,890	261,547	1,806,040	466,063
MersenneRed	340	77	280	54
ROM (enc)	10.6 kB	11.5 kB	18.7 kB	18.1 kB
ROM (dec)	6.4 kB	7.2 kB	10.5 kB	9.9 kB

The cycle counts for the 192-bit security parameter set are roughly twice as high as for the 80-bit security parameter set. This is striking since for many other cryptosystems one would expect a more significant performance drop than just a factor two. However, for our case we remark that most operations have linear complexity, except for the NTT that has quasi-linear complexity. Typically, a larger modulus usually also results in a more costly reduction algorithm. However, in this special case, the reduction is cheaper even though the modulus is larger due to the special structure of the modulus. Therefore the performance drops by less than a factor of two and encryption and decryption take $194/68$ ms on the Cortex-M0 and $10.2/3.3$ ms on the Cortex-M4.

Our implementations use large precomputed tables for the keys and constants that are required by the NTT. We store the precomputed values in 32-bit words and therefore the public key (consisting of master public key and ID of the receiver) takes $2 \cdot 4$ bytes $\cdot n$ bytes of memory. For $n = 512$ that makes up for 4 kB for the public key and 2 kB for the secret key as the secret only consists of one polynomial (user secret key). For $n = 1024$ the key sizes are doubled. The precomputed tables for the NTT also take 4 kB/8 kB for $n = 512/n = 1024$. Note that the decryption only requires the table with powers of ψ^{-1} but the encryption needs tables for both powers, ψ and ψ^{-1}. Therefore the ROM consumption of the decryption is roughly 4 kB/8 kB lower than the ROM consumption of the encryption. For encryption, the precomputed tables represent up to 88% of the overall ROM consumption ($n = 1024$, Cortex-M4). One possible way to reduce that amount is to compute the powers of ψ and ψ^{-1} on the fly. But by doing so, one can no longer apply the optimizations from [12] as they require the powers of ψ and ψ^{-1} to be stored in a bit-reversed order. Thus, reducing the memory consumption in this way implies a higher running time of the NTT.

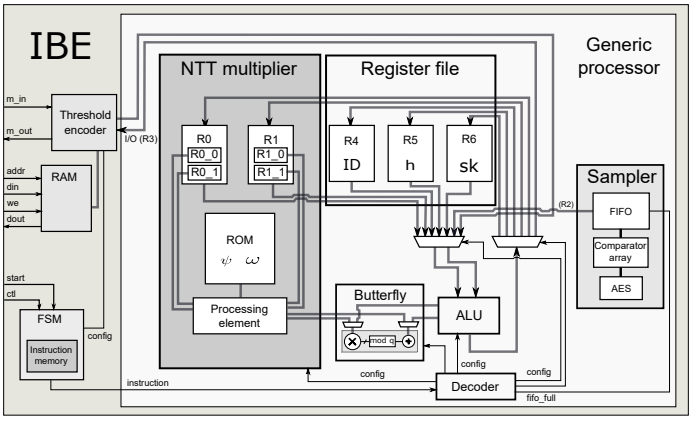

Fig. 1. Basic RLWE core of our IBE architecture, following the design of [9].

Comparison. To the best of our knowledge, this is the first implementation of a (post-quantum) IBE scheme. Therefore it is hard to find suitable implementations to compare to. One interesting observation is that of IBE and RLWE Encrypt as the encryption and decryption procedure are related. One difference between IBE and RLWE Encrypt is that the modulus is almost twice as large in terms of bit-size. For IBE, the result of a multiplication no longer fits into a 32-bit word. Therefore we have a much more costly reduction on multi-precision arithmetic.

In terms of memory consumption, our implementation for $n = 512$ is comparably large as other implementation for the parameter n. Only [21] provides significantly lower memory consumption as their implementation does not rely on large precomputed tables what make up for the major part of the memory consumption of our implementation. Our encryption routine on the Cortex-M4 for $n = 512$ is 3.7 times slower than the RLWE encryption from [21] (3.3 times for decryption). Since our modulus is larger, we cannot apply some of their advanced implementation techniques, like storing two coefficients of a polynomial in one data word. There are no suitable Cortex-M0 implementations of lattice-based schemes that we know of that we could compare our Cortex-M0 implementation to, therefore the AVR numbers of [11] and [12] are still given for reference.

TABLE III
COMPARISON OF OUR MICROCONTROLLER IMPLEMENTATIONS WITH OTHERS. IN [12] THE COMBINED BINARY SIZE FOR ENC/DEC IS GIVEN.

Implementation	Operation	Cycles	ROM/kbytes
Cortex-M0 (**this work**)	IBE enc	3,297,380	10.6
32Mhz	IBE dec	1,155,000	6.4
(512/16813057)	NTT	923,300	-
Cortex-M4 (**this work**)	IBE enc	972,744	11.5
168Mhz	IBE dec	318,539	7.2
(512/16813057)	NTT	251,763	-
Cortex-M0 (**this work**)	IBE enc	6,202,910	18.7
32Mhz	IBE dec	2,171,000	19.5
(1024/134348801)	NTT	1,749,860	-
Cortex-M4 (**this work**)	IBE enc	1,719,444	18.1
168Mhz	IBE dec	557,015	9.9
(1024/134348801)	NTT	444,117	-
Cortex-M4 ([21])	RLWE enc	261,939	1.5
168Mhz	RLWE dec	96,520	0.5
(512/12289)	NTT	73,406	-
ATxmega ([11])	RLWE enc	2,617,459	13.5
32Mhz	RLWE dec	686,367	8.5
(512/12289)	NTT	441,572	-
ATxmega ([12])	RLWE enc	2,196,945	9.2
32Mhz	RLWE dec	600,351	9.2
(512/12289)	NTT	427,827	-

B. FPGA Results

We now discuss the resource cost and performance of our implementation on a Xilinx Spartan-6 FPGA. The results of our implementation as well as a comparison with hardware implementations of RLWE Encrypt can be seen in Table IV. Similar to the comparison of software implementations, one has to keep in mind that the parameter set is different and that the modulus has roughly twice the bit-size of the modulus in RLWE Encrypt. As the micro-code engine features a generic modular reduction to support a broad range of parameters and does not exploit the binary representation of the chosen modulus, we indeed need more than twice as many cycles to encrypt or decrypt for the 192-bit security parameter set in contrast to the 80-bit security parameter set. The number of used LUTs increases by 26% and the number of FFs by 43%.

We compare our implementation to the original work [9], the highly optimized implementation of [19], and a lightweight implementation of [20]. Compared to the original work ($n = 512$), we achieve similar cycle counts even though our modulus is twice as large. But our maximum clock frequency is also lower and we need more FPGA resources. As the Spartan6 DSP blocks can handle inputs of at most 18 bits and our modulus is larger than 24 bits, we need four DSPs in contrast to just one DSP as needed in RLWE Encrypt implementations. The results of [19] are by far superior to ours, not only because of the smaller modulus, but also because their implementation is heavily optimized for the given parameter set and we built our implementation upon a generic framework. The lightweight implementation from [20] needs 25 times less LUTs and FFs but on the other side also needs 10 times more cycles for an encryption.

We want to highlight that our implementations are two orders of magnitude faster than older implementations of pairing-based IBE schemes, like [22] and [23]. Even when considering that these are implementations for a much less powerful microcontroller (ATmega128L), our performance looks clearly superior.

V. CONCLUSIONS

In this paper we gave the first evidence that recent Identity-based Encryption schemes can be implemented on constrained embedded devices. We presented implementations of a lattice-based IBE for

TABLE IV
COMPARISON OF OUR FPGA IMPLEMENTATIONS WITH OTHERS. THE RESULTS ARE GIVEN FOR ENCRYPTION (FIRST ROW) AND DECRYPTION (SECOND ROW) OR COMBINED (ONLY ONE ROW). THE CLOCK FREQUENCY IS GIVEN IN MHZ.

Implementation	Clock	(LUT|FF|BRAM|DSP)	Cycles
IBE (S6LX25, **this work**) (512/16813057)	174	(7,023 | 6,067 | 16 | 4)	13,958 9,530
IBE (S6LX25, **this work**) (1024/134348801)	174	(8,882 | 8,686 | 27 | 4)	28,586 19,535
RLWE (S6LX16, [9]) (256/7681)	160	(4,121 | 3,513 | 14 | 1)	6,861 4,404
RLWE (V6LX75T, [9]) (512/12289)	251	(5,595 | 4,760 | 14 | 1)	13,769 8,883
RLWE (V6LX75T, [19]) (512/12289)	278	(1,536 | 953 | 3 | 1)	13,300 5,800
RLWE (S6LX9, [20]) (256/4096)	144 189	(282 | 238 | 2 | 1) (94 | 87 | 1 | 1)	136,212 66,338

three target platforms, more precisely, two low-cost microcontrollers and an FPGA. We revised the outlined parameter set for IBE encryption and proposed settings that enable efficient implementation via NTT. Our results highlight that IBE is practical for IoT devices as the performance is only slightly lower than the performance of implementations of RLWE Encrypt that do not offer the simplified key management of IBE.

ACKNOWLEDGEMENT.

This work has been partially supported by European Unions Horizon 2020 program under project number 645622 PQCRYPTO and 644729 SAFEcrypto.

REFERENCES

[1] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, ser. Lecture Notes in Computer Science, G. R. Blakley and D. Chaum, Eds., vol. 196. Springer, 1984, pp. 47–53. 1, 2

[2] D. Boneh and M. K. Franklin, "Identity-based encryption from the Weil pairing," in *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, 2001, pp. 213–229. 1, 2

[3] D. Boneh and X. Boyen, "Efficient selective-ID secure identity-based encryption without random oracles," in *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, ser. Lecture Notes in Computer Science, C. Cachin and J. Camenisch, Eds., vol. 3027. Springer, 2004, pp. 223–238. 1, 2

[4] C. Cocks, "An identity based encryption scheme based on quadratic residues," in *Cryptography and Coding, 8th IMA International Conference, Cirencester, UK, December 17-19, 2001, Proceedings*, ser. Lecture Notes in Computer Science, B. Honary, Ed., vol. 2260. Springer, 2001, pp. 360–363. 1

[5] L. Ducas, V. Lyubashevsky, and T. Prest, "Efficient identity-based encryption over NTRU lattices," in *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II*, ser. Lecture Notes in Computer Science, P. Sarkar and T. Iwata, Eds., vol. 8874. Springer, 2014, pp. 22–41. 1, 2

[6] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," in *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings*, ser. Lecture Notes in Computer Science, H. Gilbert, Ed., vol. 6110. Springer, 2010, pp. 1–23. 1, 2, 6

[7] R. Lindner and C. Peikert, "Better key sizes (and attacks) for LWE-based encryption," in *CT-RSA*, ser. Lecture Notes in Computer Science, A. Kiayias, Ed., vol. 6558. Springer, 2011, pp. 319–339. 1, 2

[8] N. Göttert, T. Feller, M. Schneider, J. Buchmann, and S. A. Huss, "On the design of hardware building blocks for modern lattice-based encryption schemes," in *CHES*, ser. Lecture Notes in Computer Science, E. Prouff and P. Schaumont, Eds., vol. 7428. Springer, 2012, pp. 512–529. 1, 2

[9] T. Pöppelmann and T. Güneysu, "Towards practical lattice-based public-key encryption on reconfigurable hardware," in *Selected Areas in Cryptography - SAC 2013 - 20th International Conference, Burnaby, BC, Canada, August 14-16, 2013, Revised Selected Papers*, ser. Lecture Notes in Computer Science, T. Lange, K. E. Lauter, and P. Lisonek, Eds., vol. 8282. Springer, 2013, pp. 68–85. 1, 4, 5, 6

[10] S. S. Roy, F. Vercauteren, N. Mentens, D. D. Chen, and I. Verbauwhede, "Compact ring-LWE cryptoprocessor," in *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, ser. Lecture Notes in Computer Science, L. Batina and M. Robshaw, Eds., vol. 8731. Springer, 2014, pp. 371–391. 1

[11] Z. Liu, H. Seo, S. S. Roy, J. Großschädl, H. Kim, and I. Verbauwhede, "Efficient ring-LWE encryption on 8-bit AVR processors," in *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, ser. Lecture Notes in Computer Science, T. Güneysu and H. Handschuh, Eds., vol. 9293. Springer, 2015, pp. 663–682. 1, 5

[12] T. Pöppelmann, T. Oder, and T. Güneysu, "High-performance ideal lattice-based cryptography on 8-bit ATxmega microcontrollers," in *Progress in Cryptology - LATINCRYPT 2015 - 4th International Conference on Cryptology and Information Security in Latin America, Guadalajara, Mexico, August 23-26, 2015, Proceedings*, ser. Lecture Notes in Computer Science, K. E. Lauter and F. Rodríguez-Henríquez, Eds., vol. 9230. Springer, 2015, pp. 346–365. 1, 3, 4, 5

[13] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," in *STOC*, H. N. Gabow and R. Fagin, Eds. ACM, 2005, pp. 84–93. 1

[14] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," *IACR Cryptology ePrint Archive*, vol. 2012, p. 230, 2012, full version of [6]. 2

[15] D. Micciancio and O. Regev, "Lattice-based cryptography," in *Post-Quantum Cryptography*, D. Bernstein, J. Buchmann, and E. Dahmen, Eds. Springer, 2009, pp. 147–191. 2

[16] P. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*. IEEE, 1994, pp. 124–134. 2

[17] Y. Chen and P. Q. Nguyen, "BKZ 2.0: Better lattice security estimates," in *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, ser. Lecture Notes in Computer Science, D. H. Lee and X. Wang, Eds., vol. 7073. Springer, 2011, pp. 1–20. 3

[18] J. Solinas, U. of Waterloo. Dept. of Combinatorics, Optimization, and U. of Waterloo. Faculty of Mathematics, *Generalized Mersenne Numbers*. Faculty of Mathematics, University of Waterloo, 1999. 3

[19] S. S. Roy, F. Vercauteren, N. Mentens, D. D. Chen, and I. Verbauwhede, "Compact hardware implementation of ring-LWE cryptosystems," *IACR Cryptology ePrint Archive*, vol. 2013, p. 866, 2013. 4, 5, 6

[20] T. Pöppelmann and T. Güneysu, "Area optimization of lightweight lattice-based encryption on reconfigurable hardware," in *IEEE International Symposium on Circuits and Systemss, ISCAS 2014, Melbourne, Victoria, Australia, June 1-5, 2014*. IEEE, 2014, pp. 2796–2799. 4, 5, 6

[21] R. de Clercq, S. S. Roy, F. Vercauteren, and I. Verbauwhede, "Efficient software implementation of ring-LWE encryption," *IACR Cryptology ePrint Archive*, vol. 2014, p. 725, 2014. 5

[22] L. B. Oliveira, D. F. Aranha, C. P. L. Gouvêa, M. Scott, D. F. Câmara, J. López, and R. Dahab, "Tinypbc: Pairings for authenticated identity-based non-interactive key distribution in sensor networks," *Computer Communications*, vol. 34, no. 3, pp. 485–493, 2011. 5

[23] X. Xiong, D. S. Wong, and X. Deng, "Tinypairing: A fast and lightweight pairing-based cryptographic library for wireless sensor networks," in *2010 IEEE Wireless Communications and Networking Conference, WCNC 2010, Proceedings, Sydney, Australia, 18-21 April 2010*. IEEE, 2010, pp. 1–6. 5